FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Hierarchical Classification using hierarchical clustering: an application to Human Activity Recognition

Joel Alexandre Ezequiel Dinis

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Pedro Mendes Moreira

Second Supervisor: João Cardoso

July 20, 2018

Hierarchical Classification using hierarchical clustering: an application to Human Activity Recognition

Joel Alexandre Ezequiel Dinis

Mestrado Integrado em Engenharia Informática e Computação

July 20, 2018

Abstract

Due to the ascension of the mobile market applications it is possible to get, in real time, personal data with accurate precision thanks to the multiple sensors present on the *smartphone* such as the accelerometer and the gyroscope for example. Because a *smartphone* is a device used and carried on a daily basis, it can be used to provide information about the user's routines. Therefore, it is possible to detect when a person spends too much time siting or if she practices a lots of sports, which can hold interesting potential since we can use the information collected by the sensors and detect if a person has harmful behaviors and warn her about its danger. In example, around 60% of global deaths are caused by chronic diseases such as diabetes, cardiovascular diseases and even some types of cancer. It is thought that sedentary behavior plays a major role in the development of these conditions. We could prevent most of these problems, with one simple warning on the mobile phone "You have been inactive for long periods of time. Remember to move once in a while", if it would be possible to know the daily routine of the user. But this is not the only benefit to be extracted with this type of information. It is also possible, to, knowing the everyday life of the user recommend her content (movies,stores) for activities she is interested on. Therefore, it is this work's job to create and implement hierarchical clustering techniques that are able to detect activities with good precision, using machine learning techniques, as well as datasets, for training and testing models, and using an hierarchical approach. The project is based on an existing project developed by INESC-TEC which provides background *framework*, which allows us to abstract some more technical concepts. However, this approach proved to fall short to multi classifier approaches. For the first experience the implemented technique achieved an accuracy of 45%, while both Hoeffding Tree and Naive Bayes got an accuracy of around 70%. The KNN results were 14%, for reasons that will be explored further in the work. The same happened for the second experience where the implemented technique achieved an accuracy of around 64.5% and both Hoeffding Tree and Naive Bayes achieved an accuracy of 88% while the KNN was the worst yet again, with an accuracy of 33%.

Keywords: Human Activity Recognition, Hierarchical clustering, Hierarchical classification

Resumo

Todos os dias carregamos smartphones, relógios inteligentes com bastantes sensores, que poderiam ser usados para melhorar a qualidade das pessoas. Se fosse possível averiguar a atividade que um utilizador está a executar, seria possível salvar muitas vidas humanas, pois uma das maiores causas de morte deriva de uma vida sedentária. Os utilizadores cujo grau de imobilidade fosse alto, seriam avisados deste fato para se movientarem, a fim de evitarem futuras complicações cardíacas, entre outras, causadas pelo estilo de vida sedentário já referido. Claro que há inúmeras aplicações, desde espionagem a publicidade direcionada. Para isso, já há bastantes estudos feitos na área de reconhecimento de atividades humanas. Contudo, não há uma abordagem que seja fiável a 100%, e bastantes delas apenas funcionam em ambientes controlados, sendo que à passagem no ambiente real falham em cumprir o objeto. Foi assim, proposto, um estudo que tentasse uma abordagem hierárquica com recurso a técnicas de clustering hierárquico, baseado mais numa abordagem divisiva em prol da abordagem aglomerativa que usualmente é mais usada por ser computacionalmente menos exaustiva. Contudo, com poucos elementos, é menos exaustiva do que a aglomerativa e é passiva de sofrer melhoramentos que podem condicionar e muito a sua performance. Com base em artigos já elaborados nesta área, nomeadamente o [KR09] na qual foi baseado o algoritmo divisivo, foi então elaborada uma tentativa de resolver este problema do reconhecimento das atividades humanas. Contudo, como se pode comprovar em capítulos mais abaixo, a técnica ficou aquém das expectativas, tendo sido ultrapassada por técnicas multi classificativas, sendo que apenas conseguiu uma eficácia de 45%, com os classificadores Naive Bayes e Hoeffding Tree a conseguir uma eficácia de perto de 70% e com o KNN com 14.1%. O mesmo aconteceu à segunda experiência onde os classificadores Naive Bayes e Hoeffding Tree conseguiram uma eficácia de cerca de 88% , 33% com o KNN, e esta abordagem implementada conseguiu uma eficácia de 64.5%.

Keywords: Reconhecimento Atividades Humanas, Clustering hierárquico, Classificação hierárquica

"The moment you are not afraid is the moment you should stop doing it"

Maximilian Berger

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations

FEUP	Faculty of Engineering of University of Porto
KNN	K-Nearest Neighbor
MOA	Massive *Online* Analysis
RAM	Random Access Memory
CPU	Central Processing Unit
HAR	Human Activity Recognition
DIANA	Divisive Analysis Clustering
SVM	Support vector machine

Chapter 1

Motivation and Objectives

1.1 Motivation

The world we live is a world dominated by technology. Wherever we look, we are surrounded by this new reality and it shows no signs of slowing down. One example of this new mundane reality is the *smartphone*, which is essential in our day to day routine allowing us to make calls, finding places using online maps with the help of GPS or even allowing us to play a game to entertain ourselves among other equally interesting tasks. All of this potential was soon discovered by a large majority of people and so almost everyone in the world has one. This fact raises a very important question, could we use the fact that people take their *smartphones* everywhere to their advantage? Because of the multiple sensors available in one of these devices it is possible to make an estimate of the activity being performed by the user. This can prove to be very useful. One of the major causes of chronic diseases is not exercising enough. Imagining we could warn the user about the fact that he is not moving enough during his routine his chances of developing this disease would be reduced [EPMK08]. Another use of this technology would be targeted advertisement. Exemplifying this to a more concrete extent if it would be possible to detect that the user is fond of running it is possible to recommend running content, or if it would be possible to detect that the user likes cooking, content could be sent to him related to this activity. Studies have been made in this area of knowledge, with more focus on some activities trying to figure out a solution to this problem. This study will, therefore, approach some of their methods, results and problems found and incorporate their findings in our work, and use Divisive Analysis in order to study its possible effectiveness on the presented problem, as well as some advantages or disadvantages it might have compared to other approaches. This approach will be compared to a normal multi-class classification in order to have a level of comparison between the two.

1.2 Proposed Solution

Given that there still hasn't been found a solution for recognizing human activities that can be it 100% trustworthy, is expected the development of a new algorithm or technique that can recognize activities. It has been referenced in papers, like in [XMZT17] that hierarchical classification usage can lead to an increase in accuracy compared to multi-class classification approaches. To get measures on how distant/close the classes are, a divisive analysis clustering technique will be implemented, based on the work [KR09]. In the previously mentioned work, it is described a clustering technique that will be incorporated on the developed project. A tree will be generated by this algorithm, and then be tested and see how well it matches with the multi-class classifiers. A deep analysis on the approach is made as well as a interpretation for the results.

1.3 Document structure

As such, this document will be divided into five chapters. The first one is focused on introducing the problem, the setup which we are presented with. The second one will approach the solutions explored by other studies, trying to elaborate on their works with the maximum detail. The next one will tackle our own approach to this problem, exploring our own ideas, methods and thoughts about this problem and how we can contribute to the exploration of this field of studies. Analyzing and interpreting the results obtained will be the subject of study of chapter four. Finally, on the fifth chapter a small conclusion about the gained knowledge as well as future work will be made.

Chapter 2

Bibliographic review and Solution Context in State of Art

2.1 Introduction

In this chapter we will review some works related to this field of studies and introducing some important concepts and methods relevant to the objective of the work. As we previously saw, the field of studies mentioned is potentially useful as it can prove to be extremely useful in areas such as health care, advertisement and emergency detection. We can use this technology to study the routine of the user and detect scenarios where we can intervene to help and improve the quality of life of the user. Examples of this are chronic diseases such as diabetes, cardiovascular diseases or osteoporosis. They are cause of 60% of the world's total deaths and these have a deep root embed in the lack of practice of physical exercise [EPMK08]. Therefore, if we could somehow detect lack of exercise from the user, we could warn him about this danger. But this is just one example of many it is possible to formulate .One of the characteristics it comes to mind when thinking about this field of studies is its dynamic component. In fact, we cannot assume that every person does an action the same way another person does it. This is because every person has different way of moving, different way of moving their arms and even physiological aspects such as height, weight, gender alone can influence results. Because the potential for the development of this technology has been demonstrated we are now able to proceed further in the exploration of this subject.

2.2 Data extraction through sensors

In order to decipher the action the user is performing, firstly we must explore the tools that will help us accomplish this task. By measuring certain sensors embedded in the *smartphone*, we are able to create an assumption on the activity of the user. Depending on the phone, it will have more or less sensors which can helps us track his activity story. Sensors can be divided into four

groups according to their category of use: environmental, acceleration, location and physiological [LL13].

2.2.1 Environmental sensors

Sensors capture the state of the environment according to the information received. Such sensors include humidity sensors, light intensity sensors, thermometers. These include just some of the existing sensors in current *smartphones* that can help the activity of recognizing human activities. There is also video and sound capture, however, because of their high computational complexity, they are not considered as a primary source of information.

2.2.2 Acceleration

Triaxial accelerometers are widely used in this field of research. They are cheap, have high accuracy and do not use a lot of computational power. These type of sensors detect the movement of user in the 3 axis X(lateral), Y(vertical), Z(longitudinal) and are critical in detecting activities such as *running*, *walking*, *cycling* or *standing still*. They can also be used in complex activities such as *eating*, *playing poker*, *writing*, *drinking* due to their ability to read such body movements, as long as the sensor is placed accordingly to the activity to be predicted. Although these activities are prone to being similar, more sensors are needed to eradicate false positives. Accelerometers are usually partnered with gyroscopes as they allow to create an estimate on how much the position of the *smartphone* changed.

2.2.3 Location

Location can be useful to accurately extract user activity. To better contextualize, picture a situation where the GPS reveals the user is located on the street. As such, it is unlikely he will be sleeping on the street. Introducing these limitations in our software it is expected the error rate gets lowered, allowing for a better performance overall.

2.2.4 Physiological sensors

These sensors are not usually incorporated on *smartphones*. They rely on reading body signs like body temperature, heart rate, among others. Although they add value to these systems, as said before *smartphones* do not usually possess this kind of sensors.

2.3 Supervised Learning

This approach is based on data, usually in form of datasets, based on data labeled by humans. The algorithm is trained using labeled data which requires a human to confirm the activity being performed which is then capable to create a model to predict future activities.

2.4 Semi-Supervised Learning

It is characterized by the use of both labeled and unlabeled data. Labeled data is used to create a model to predict the user's activities using the same line of thinking as the approach above mentioned, although it uses unlabeled data to improve its algorithm constantly to adapt the best to its user. This is an interesting way to tackle the problem because unlabeled data is very easy to collect and it doesn't require human interference. The problem with unlabeled data is that in most situations, it is expensive to obtain the labels. It does not necessarily require humans to label it.

2.4.1 *Online* and *Offline* approaches

Both semi-supervised learning and supervised learning can be either *offline* or *online*. The difference between these is, while the *offline* version of this problem only identifies activities after a period of time, the *online* version tries to recognize the activity in real time. One example of this given in [LL13] would be the applications that count the number of calories burn after a set of exercises or targeted advertisement, since we would do a complete study on the user's routine so we could provide him with advertisements related to his interests.

2.5 Hierarchical classification

Through Hierarchical classification one instance must be classified several times, while moving in the generated tree in order to get the classification, once reacher the leaf node, instead of comparing this instance with others right away(multi-class classification) it is done a pre selection of similar instances a group can be created with and so there are several layers to determinate a solution. In the context of our problem it means that we will aggregate activities as shown in Figure 2.1 from the more generalized group to the specific activities. In each step and according to the activities, classifiers will be executed and in each level of the tree, a choice is made according to the classifier result. For each level, it is expected that the best classifier to distinguish activities is chosen for the maximum efficiency of this algorithm. This approach to the problem is relatively new, with some promising results in some studies [Zhe15]. In order to make an hierarchical classification it is needed to know the proximity or distance between classes as well as which classifier separates the activities better. To get such information clustering is done. There are two possible ways when using clustering (Agglomerative Clustering and Divisive Clustering). In our case, the latter was chosen due to scientific neglect of the scientific community due to its computational requirements (although a lot of the required divisions can be avoided). The context of the project was also another key factor in the decision, since it makes more sense to go from a node with multiple classes and then have all of them separated in the end. The agglomerative approach consists on the opposite, where a lot of clusters generate in the end one bigger one.

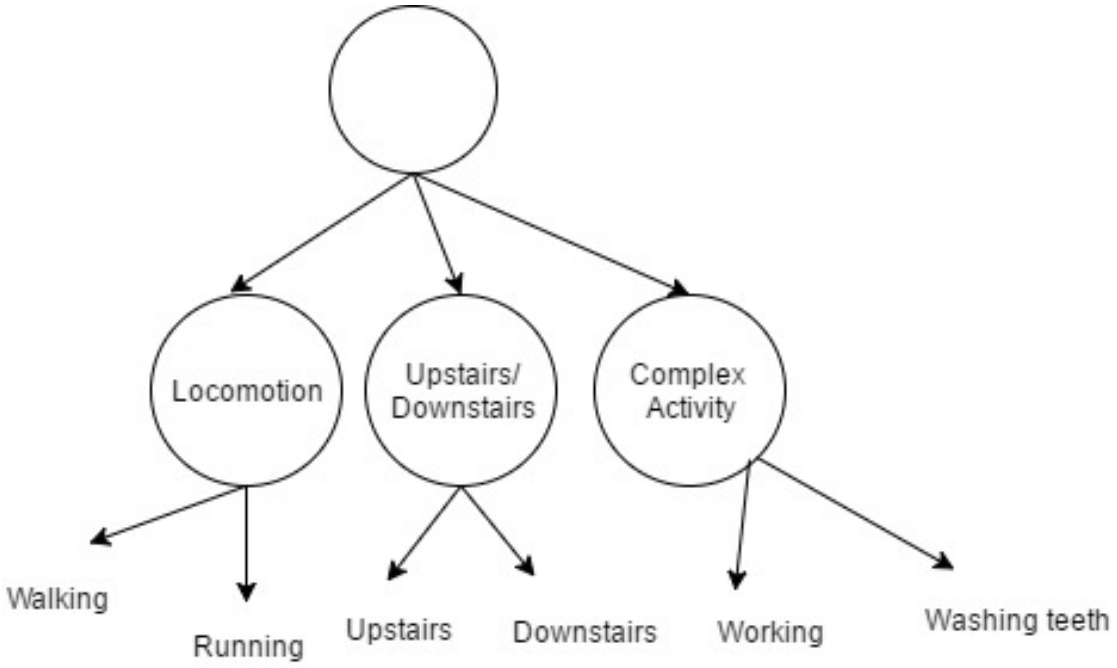

Figure 2.1: Example of hierarchical subdivision

This project is widely based in hierarchical clustering, more specifically DIANA, described in section 2.5.1. What this means is that we begin with one big cluster which gathers all activities to be predicted and then, based on several variables, we keep separating them, according to their degree of distance, until we obtain them separately.

2.5.1 DIANA

DIANA stands for DIvisive ANAlysis Clustering, a hierarchical clustering top-down approach. Although there is also an agglomerative approach that could be used to solve this problem, after some discussion it was decided to implement the divisive approach it would make more sense to use the first one, since it is more natural to start with one big cluster of activities and in the end obtain them separately than to start with this activities separately. While DIANA can be computationally more expensive, a lot of divisions can be avoided, and because this approach, has said before, was not very used due to the computational requirements, can possibly reveal itself promising. It is important to understand this in order to completely understand the work developed. First, it is important to find a splinter group. To find this, it is firstly necessary to find the member that is the most different to the others. DIANA is widely studied and reviewed in [KR09]. In this study, the author introduces this clustering technique. To begin executing this technique, it is necessary that a matrix of dissimilarities is calculated. This is where things can get tricky, since to do so, there are a number of criteria such as Euclidean distance, the Manhattan distance, Mahalanobis distance among others. The final results will depend on the criterion picked.

To discover which element is the most different from others, we calculate a variable called Average Dissimilarity, which is calculated by making an average of dissimilarities among all the

	a	b	c	d	e
a	0.0	2.0	6.0	10.0	9.0
b	2.0	0.0	5.0	9.0	8.0
c	6.0	5.0	0.0	4.0	5.0
d	10.0	9.0	4.0	0.0	3.0
e	9.0	8.0	5.0	3.0	0.0

Figure 2.2: Example of matrix of dissimilarities

groups.
N - Last activity/element
K - Number of activities/elements
L - Line that is being calculated Average Dissimilarity

$$AverageDissimilarity(L) = \frac{\sum_{i=b}^{N} f(L,b)}{K-1}$$

Basically, to calculate the average dissimilarity of a line it is needed to sum all the elements from the line with f(L,a) until the last element f(L,N) divided by the number of activities minus one. To exemplify with the previous example,for the first line, it would come f(a,b), f(a,c), f(a,d), f(a,e), sum the results and divide by the number of activities minus one.

Object	**Average Dissimilarity**
a	(2.0+6.0+10.0+9.0)/4= 6.75
b	(2.0+5.0+9.0+8.0)/4= 6.00
c	(6.0+5.0+4.0+5.0)/4= 5.00
d	(10.0+9.0+4.0+3.0)/4= 6.50
e	(9.0+8.0+5.0+3.0)/4= 6.25

Table 2.1: First iteration of DIANA split group

As we can see in table 2.1, the element that disagrees the most with the others is element a. What this means is that, in the cluster {a,b,c,d,e}, the previously mentioned element is the one who has the most differences to the others. However, this iteration is not finished, as now we can test which elements agree more with element a than with the remaining group. As such, now that we have divided the initial cluster into two temporary ones, {a} and {b,c,d,e} we will iteratively continue the division.

Basically what it is being done is that we are comparing if each member agrees more with the cluster it is already with, or if it agrees more with the newly created splinter group. In table 2.2, we can observe that the only element that agrees more with the splinter group than the other group is element b, where we can see a positive difference. Therefore, a new element is added to the group and so we have two temporary clusters consisting of {a,b} and {c,d,e}. However, since we

Object	Average Dissimilarity	Average Dissimilarity to Splinter Group	Difference
b	(5.0+9.0+8.0)/3= 7.33	2.0	7.33 - 2.0 = 5.33
c	(5.0+4.0+5.0)/3= 4.67	6.0	4.67 - 6.0 = -1.33
d	(9.0+4.0+3.0)/3= 5.33	10.0	5.33 - 10.0 = -4.67
e	(8.0+5.0+3.0)/3= 5.33	9.0	5.33 - 9.0 = -3.67

Table 2.2: Second iteration of DIANA split group

did not obtain a situation where all the differences in the table are negative we continue with the iteration.

Object	Average Dissimilarity	Average Dissimilarity to Splinter Group	Difference
c	(4.0+5.0)/2= 4.50	(6.0 + 5.0)/2 = 5.50	4.50 - 5.50 = -1.00
d	(4.0+3.0)/2= 3.50	(10.0 + 9.0)/2 = 9.50	3.50 - 9.50 = -6.00
e	(5.0+3.0)/2= 4.0	(9.0 + 8.0)/2 = 8.50	8.50 - 4.0 = -4.50

Table 2.3: Third iteration of DIANA split group

Analyzing the table 2.3 it is possible to see now that there is no elements that agree with the splinter group. As such, the division of the cluster has the final results of {a,b} to {c,d,e}. Still, this technique has not reached an end. With this complete iteration a level of the tree has been unfolded, however, there are still groups which are not completely separated. The {a,b} will obviously divide itself into {a} and {b}, but we do not know how the cluster {c,d,e} will be divided. If there is no clear path of which cluster to divide next, the cluster to be chosen is the one with the biggest diameter. In this particular example, the diameter of {a,b} is 2.0 while the other cluster diameter is {c,d,e} 5.0. As such, the latter is chosen. So now it is imperative do divide the cluster {c,d,e} in an optimal way to express the difference between the belonging elements. Since the cluster only have the c,d,e elements the corresponding matrix will only include these elements.

	c	d	e
c	0.0	4.0	5.0
d	4.0	0.0	3.0
e	5.0	3.0	0.0

Figure 2.3: Matrix for cluster {c,d,e}

Again, it is needed to find the splinter group for the matrix.

Analyzing the table 2.4, it is visible that the element who most differs from the others is element c and, as such, it is added to the splinter group. However, it is still not clear if any of the other elements will join c in the splinter group.

Object	Average Dissimilarity
c	(4.0+5.0)/2= 4.50
d	(4.0+3.0)/4= 3.50
e	(5.0+3.0)/4= 4.00

Table 2.4: First iteration of DIANA for cluster {c,d,e}

Object	Average Dissimilarity	Average Dissimilarity to Splinter Group	Difference
d	3.0	4.0	3.0 - 4.50 = -1.00
e	3.0	5.0	3.0 - 5.0 = -2.00

Table 2.5: Second iteration of DIANA for cluster {c,d,e}

Since there are no positive differences, none of the remaining elements agree more with the splinter group than the other group. As such, the process ends and the final clusters for the second level of the tree are {c} and {d,e}. It is obvious now how the {d,e} will divide, but for the purpose of showing the algorithm it is imperative to continue the process. Since the cluster to divide is {d,e}, only the elements d,e and are relevant to the current iteration. Therefore, the matrix associated is presented below:

$$\begin{vmatrix} & d & e \\ d & 0.0 & 3.0 \\ e & 3.0 & 0.0 \end{vmatrix}$$

Figure 2.4: Matrix for cluster {d,e}

The Average Dissimilarity has the same value for both elements, so we can pick one at random, in example, element d. However, it is not possible for element e to join that group, since there are only two elements remaining. Because of the previously mentioned fact, the iteration stops, owning to the fact that there are no other clusters composed by multiple elements as the cluster {a,b} is divided as {a} and {b}. It is possible now, to see the tree that comes as a result produced by this technique.

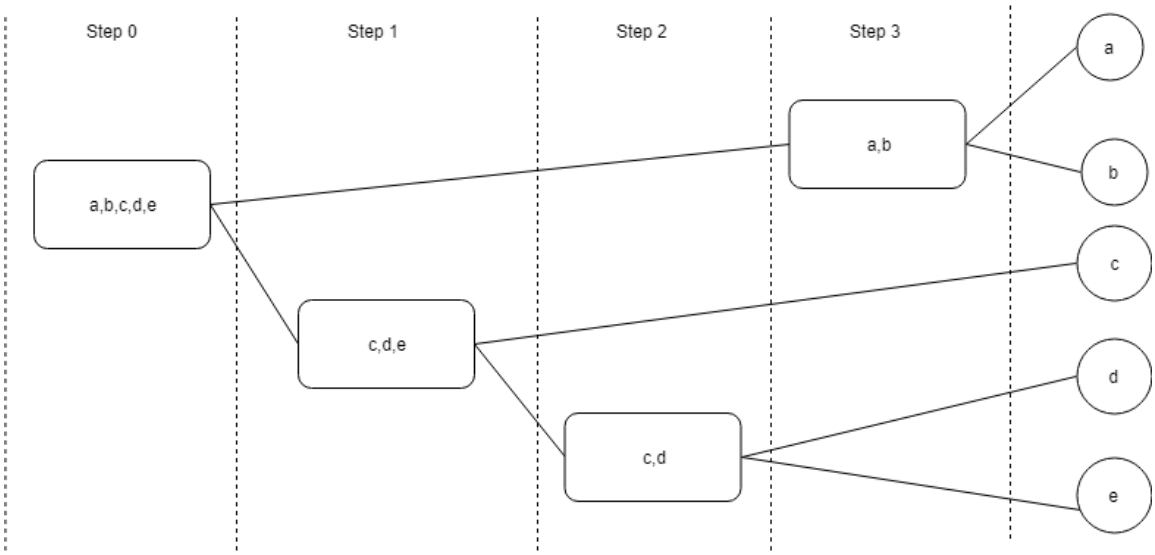

Figure 2.5: Tree resulting from DIANA

2.6 Classifiers

In order to better understand the process, it is also necessary to understand the algorithm behind the classifiers. As such, a brief introduction to them is done.

2.6.1 Naive Bayes

Naive Bayes is one of the most used algorithms in machine learning. It is highly scalable and relatively simple being able to outclass more complex algorithms with the correct pre-processing. Returning to our context what we will be given are determinate features like variance, mean and with this information the algorithm will calculate the probability of one activity corresponding to one determinate label and pick the most probable answer.

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Figure 2.6: Formula for Naive Bayes

The algorithm makes a class prediction, using probabilities, calculated using the data provided.

2.6.2 K-Nearest Neighbor

Another common machine learning method. It uses known points which already have a classification to provide an estimation on where a new point belongs. Let's consider one simple example.

Imagine, that there exist two possible classes to classify one object, A or B. So, in our world of possibilities it is only possible to classify objects with the A class or the B class, being one class properly defined with specific attributes. For a new instance, named C, we must figure out what to classify it as. The K closest values to the new instance are the ones that decide the classification of the new point. In figure 2.7 we can see this situation represented. This is a very simple and generic method in which we simply classify one example according to its K nearest neighbors (for which we know the classes).

Figure 2.7: K-Nearest Neighbor

2.6.3 Decision trees

The main characteristic of decision trees is the choice at each level(relative to every branch division). In each, we are presented with a separate direction. This, although, is not done randomly as it is calculated which possibility gives us the best gain for which branch division. To give an example, imagine we are classifying an instance C (new point) given the existence of some classes A, B, F and E. Imagine also that the depending variables would be mean and angular velocity for better contextualization of this problem. In figure 2.8 this problem is described into a graphical visualization. To notice that in this graph it is not shown the calculation of the optimal branching algorithm which involves calculation of gains and exploring the possibility of different branching. However, one more even interesting concept is the very fast decision. With a concrete labeled dataset is able to iteratively learn which nodes to expand to get better results, testing which ones are more effective to split. The performance of this very fast decision tree is similar to a regular one decision tree.

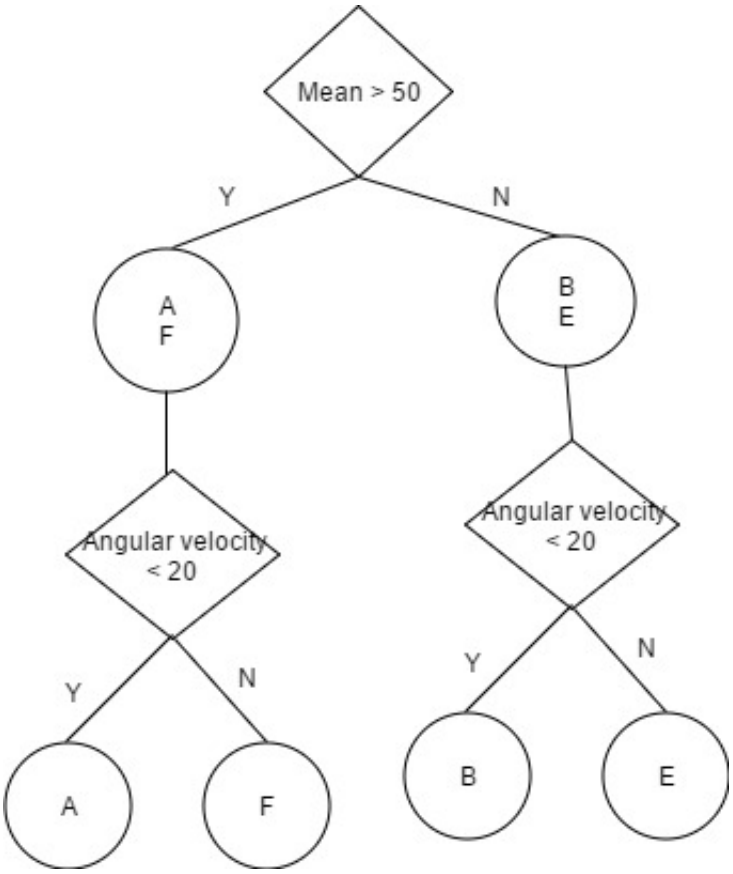

Figure 2.8: Decision tree example

2.7 Sliding Windows

Recognizing activities has been a very extensive subject to describe. However what we are not considering yet, is the window of capture of an activity. As we will see there are different options for this action referred by [Car16] which will be studied carefully.

2.7.1 Non Overlapping Sliding Windows

As the name itself suggests, it is a very simple way to capture activities from the sensors. All it is done is specifying a window size, and according to the number of received samples, they will never overlap each other. It is needed to add that in this method every window has a static length. For a proper example, if the window size is 2 and we have 6 samples, each window will have 2 consecutive samples like 1,2 and other 3,4 and so on.

2.7.2 Overlapping Sliding Windows

Again the name is very revealing of the function of the windows. Unlike the previous method where it was not possible to have same samples on different windows, this becomes possible.

Therefore it is possible that, a window with a size of 2 has sample 1 and 2 but another window could have sample 2 and 3. In conclusion, it basically means that it overlaps windows in order to get a more accurate measure. This is the used sliding window on the project.

2.7.3 Dynamic Signal Segmentation Windows

Some new approaches have been proposed these last years, and so, it was questioned if the windows could have a dynamic length overcoming in this way one of the main disadvantages of the previous presented models, since they do not respect the fact that a window can have more than 1 activity present.

2.8 Data sampling

Because the *smartphone* has memory limitations it is necessary to select the most important/latest data to arrive so we can keep in store the most relevant information. In [dOL12] two types of methods are presented: sequence-based windows and timestamp-based windows.

2.8.1 Sequence-based windows

The first thought that crosses our minds when thinking about getting the last data is simple. Just keep storing the latest information and when the memory starts to become overwhelmed we will ditch the oldest data sample. This is what the main core of the algorithm consists of, in simple words.

2.8.2 Timestamp-based windows

Another set of algorithms are the timestamp-based windows although the biggest representative of this class is the "priority sample". It simply consists on defining a interval T of time and assigning for each data arrival a random priority. The data is included in the sample if the interval T has not been reached yet and there is not a data whose interval T has also not been reached and it has less priority than the one which we want to include.

2.9 Feature extraction

In order to recognize different activities firstly we need firstly to think about the process of identification of one activity. Well, it was mentioned above we would recur to sensors for information gathering. However this raises one problem: how can we extract information from these sensors? Well, some sensors are easy to use. Such case is the use of GPS. Through this sensor we know the coordinates of the user without effort. How do we however extract information from sensors like accelerometer, gyroscope which output a signal? In [FDFC10] this problem is addressed, and the authors propose three ways to extract information from the waves:

- **Time domain:** Basic mathematic and statistical analysis. Involving few computational cost, and relatively simple operations they have a multitude of uses. One example is the mean, since it can be used to identify user posture. Another example is the usage of standard deviation to create an estimate of the stability of the signal.

- **Frequency domain:** Used to recognize repetitive patterns in a signal since this repetition is usually connected to several activities like walking or running [FDFC10]. Example of this technique is the analysis of the spectral energy which can reveal the way of transport a user is using (driving, cycling, walking).

- **Discrete domain:** Technique involving the transformation of signals into strings of discrete symbols. One example of this technique is the calculation of the Euclidian distance which provides a fast discrimination of signals. Another particular interesting variable is the Levenshtein edit distance. This variable is able to identify a number of gestures. In [FDFC10] is also referred a study managing to get 83% accuracy on the the previous activity using this distance metric.

Time Domain	**Frequency domain**	**Discrete domain**
Mean Median	Spectral Energy	Euclidean-related distances
Variance and Std Deviation	Information Entropy	Minimum distance
Root Mean Square	Dominant frequency	Dynamic time warping

Table 2.6: Some features of each domain

2.10 Related Work using Hierarchical classification

The project developed by these scientists described in [QMXW10] was based on the premise that, as seen in [NJ02] discriminative classifiers perform better than generative classifiers and as such it was proposed to build an hierarchical classification model using SVM approaches for the activities walk, jogging, fall, stand to sit, stand to squat and in-place actions. When collecting the results from the tree and comparing to other approaches pointed in the study, it was the approach with the best accuracy with 88.69%. In [BDP+13], an hierarchical approach fused with majority voting technique is used. While a tree is used as the main structure of the project, multiple classifiers are used per branch giving each one of them different weighting power. However, at the time of the final classification, all the information is used to make a decision. The approach achieved some very promising results. In [XHZ+18] an approach to create an hierarchical decision tree based on different components of the instances(frequency, amplitude, orientation) was made. It was also taken into account the context of the hierarchy(in example, walking is normal to appear succeeding a running action) and it was built several transition trees to represent this. The approach produced interesting results managing the 96.68% accuracy. Depending on the selected features, a class can be classified as similar or dissimilar. The problem is that this can happen for the same class but

picking different features to evaluate them by. The study [MRS12] tackles this problem, while using an hierarchical approach in junction with the previously mentioned technique to recognize human activities. Similarly to the previously presented study by Xu et al, [CWZ+17] was used a context based approach, where only some activities can proceed others. Based on this premise, it is possible to build a tree each state represents one activity and only certain nodes are connected. Compared to all the other classifiers used alone, the solution developed was able to be the best.

2.11 Evaluation

There are several ways to evaluate the results obtained. In this study, Holdout and Leave-One-Out were used. As such it is required to define both of the concepts for a better understanding of the work. Although they can both be used as parallel techniques to evaluate separately, in this project they were used together.

2.11.1 Holdout

The holdout method is one of the most simple validation methods. Basically one sample is picked to test the model generated by the training set randomly. The holdout evaluation was used for the cluster analysis, since a user had to be picked in order to proceed with the information extraction.

2.11.2 Leave-One-Out cross validation

The leave one out technique consists in creating a model with the training dataset, and leaves one sample out to do the testing part. In this study a user is left out as a sample, since every activity needs to be present on the testing dataset in order to collect results for it. The user to be tested is rotated for the N existing users, and an average of all the collected results is calculated in order to obtain a more realistic representation of the results. After the tree, which was obtained with the cluster analysis, using the holdout method, it is now the time to test the tree created with this analysis. To do so, the leave one out cross validation is now used. It can be said, that, in this work they complement each other.

2.12 Collection of results

With the data retrieved from the classifiers, it is possible to calculate a *confusion matrix*.

Activity/Identified As	**Running**	**Brushing Teeth**	**Riding Bicycle**
Running	60	5	0
Brushing Teeth	4	50	2
Riding bicycle	0	2	48

Table 2.7: Example of *confusion matrix*

Using this concept, it becomes possible to estimate the accuracy, precision and other relevant variables with the objective of figuring out if our approach provides a good solution. The **accuracy**, for instance, provides a general classification performance for all the activities.

TP:"Positive" activities that were classified as positive.
TN:"Negative" activities that were classified as negative.
FP:"Negative" activities that were classified as positive.
FP:"Positive" activities that were classified as negative.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Providing another important measurement variable is the **precision**. It provides information on the percentage of activities that are correctly predicted, given they were identified as positive.

$$Precision = \frac{TP}{TP + FP}$$

However, there are more useful measures such as **recall** or the **f-masure**. Respectively, the **recall** variable provides the percentage of "positive" activities that were correctly identified. **The f-measure**, combines both the **precision** and **recall** value.

$$Recall = \frac{TP}{TP + FN}$$

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Chapter 3

Proposed Method

Based on the explained concepts on the previous chapter, it is now the time to demonstrate what was developed to solve the problem. It is important to understand that this project is basically the continuation of an existing one from INESC-TEC. This provides an abstraction to some concepts which are hard to implement, nonetheless crucial for the implementation described in the document. The approach used is based on the DIANA technique presented in [KR09], however it is a modified version in order to explore how it can be improved. Firstly, instead of using Manhattan or Euclidean distance, among others, it was created a new measure, different from the previously used. Secondly, instead of using only the DIANA technique for one only classifier, three classifiers were used which allows for maximum accuracy. In the sections below however, everything will be explain more clearly.

3.1 INESC-TEC

As it was described before, the solution that was implemented was based on an existing project from INESC-TEC. The purpose of this project was to, given a dataset with many activities, discover how well a multi-classifier solution, using Naive Bayes, Hoeffding Trees and KNN, could predict activities, on a semi-supervised learning environment.
To do so, however, there were a lot of concepts necessary to incorporate in the INESC-TEC project, such as overlapping windows, as well as the previously mentioned semi-supervised learning component or even the extraction of appropriate features from the sensors data, among others. To do so requires a lot of time, and as such, the previously existing project was used as a framework for the new one, since they both share similar requirements.

3.2 Approach

In the following section an in depth description of the used methods will be done, introducing the distance criterion used, as well as the integration with the DIANA technique alongside a review of the dataset, architecture and used tools. To be able to cluster the data and get an analysis that can enable distinction between classes, it is done a multi-class classification(using the classifiers described in other sections), by making holdout of user 6(since it had a wide variety of instances from all classes). With said results, a confusion matrix is built. However, in order to run the DIANA technique, a dissimilarity matrix is needed and, as such, a process needs to be made to extract it from the confusion matrix. To do so, a distance technique needs to be defined. With this, it is possible to execute the DIANA algorithm, and collect the results afterwards. The experiments protocol will also be presented.

3.2.1 Distances and Dissimilarity Matrix

With the previously implemented setup, and using the PAMAP dataset, which will be more properly described in the sections below, as well as configuring important values at the start such as, the overlap percentage, the length of the windows (consult subsection 3.8.1 for more information), the classifiers to be used, and which sensors' data to use for the classification. After the environment has been created, the classifiers are trained with data from users for the generation of the model, and it is based with this model that instances will be classified and the result will be used for the modified DIANA algorithm. It is important to note that users are left aside when doing the training phase, so there are users which were not a part of this phase, to do the testing(again refer to chapter 4 for more detailed information). So, when the testing phase ends it is possible to collect results. As such, methods were created in order to receive a confusion matrix for each classifier used. The used classifiers were the KNN, Hoeffding Tree and Naive Bayes, which were the ones available in the inherited project. What a confusion matrix represents is all the possibilities for all activities. To provide a more concrete example, the activity[0] that was actually classified as 0 is 540. Alongside the line, it is possible to see all the activities that were 0, but were instead classified as activity[1],[2],etc. This is how a confusion matrix should be read. This is why diagonals have such big numbers, since these represent the correct instances.

$$\begin{bmatrix} 540 & 0 & 10 & 0 & 0 & 10 & 0 & 16 & 0 & 5 & 2 \\ 0 & 404 & 123 & 0 & 0 & 1 & 0 & 4 & 2 & 14 & 28 \\ 0 & 169 & 317 & 0 & 0 & 2 & 0 & 3 & 0 & 27 & 91 \\ 0 & 0 & 18 & 401 & 2 & 53 & 88 & 49 & 4 & 18 & 2 \\ 0 & 0 & 17 & 55 & 367 & 0 & 91 & 3 & 19 & 0 & 19 \\ 0 & 11 & 5 & 0 & 0 & 406 & 0 & 12 & 3 & 58 & 13 \\ 0 & 0 & 16 & 175 & 13 & 6 & 444 & 3 & 3 & 1 & 2 \\ 0 & 10 & 30 & 15 & 1 & 53 & 7 & 147 & 4 & 25 & 13 \\ 0 & 24 & 10 & 46 & 9 & 1 & 14 & 18 & 139 & 12 & 5 \\ 0 & 0 & 56 & 27 & 1 & 36 & 5 & 66 & 32 & 199 & 102 \\ 0 & 70 & 239 & 0 & 0 & 1 & 0 & 3 & 3 & 109 & 519 \end{bmatrix} \tag{3.1}$$

Figure 3.1: Example of confusion matrix for Hoeffding Tree

When the confusion matrix is calculated for each classifier, the next step is to calculate the dissimilarity matrix for each one as well. This step, instead of using the already previously mentioned distance algorithms, like Manhattan or Euclidean between others, uses a customized one. This customized measure one activity with every other to inference how easy to differentiate them. To give a concrete example, based on the previous matrix, consider the lines one and three.

$$\begin{bmatrix} 540 & 10 \\ 0 & 317 \end{bmatrix} \tag{3.2}$$

Figure 3.2: Example of distance calculation between lines one and three

To calculate how well the two classes are distinguished, it requires just a calculus regarding the correct instances and total instances. The values from lines one and three extracted from the confusion matrix, from their respective indexes. Due to the fact that the confusion matrix represents the relation prediction/real class, we can deduce that if the activity of index X, predicts that the activity is X, then the activity is correct. It is then no surprise, that alongside the diagonal of the matrix there are only big numbers, since this number represents the number of correct activities. Returning to this example, we now calculate the dissimilarity between the two which is $(540+317)/(540+317+10) \approx 0.98$. The maximum value a dissimilarity can achieve using this measure is 1.0, while the minimum is 0.0. The value 1.0 means that the two activities are perfectly distinguishable while a value of 0.0 means they cannot be distinguished at all. By applying this technique to all the activities, a dissimilarity matrix is created, for each classifier. Taking the example using Hoeffding Tree above and applying this concept we get the results below. Note the results are approximate, since the real values do not fit the entire width of the page.

$$\begin{bmatrix}
0.0 & 1.0 & 0.98 & 1.0 & 1.0 & 0.98 & 1.0 & 0.97 & 1.0 & 0.99 & 0.99 \\
1.0 & 0.0 & 0.71 & 1.0 & 1.0 & 0.98 & 1.0 & 0.97 & 0.95 & 0.97 & 0.90 \\
0.98 & 0.71 & 0.0 & 0.97 & 0.97 & 0.99 & 0.97 & 0.93 & 0.97 & 0.86 & 0.71 \\
1.0 & 1.0 & 0.97 & 0.0 & 0.93 & 0.93 & 0.76 & 0.89 & 0.91 & 0.93 & 0.99 \\
1.0 & 1.0 & 0.97 & 0.93 & 0.0 & 1.0 & 0.88 & 0.99 & 0.94 & 0.99 & 0.97 \\
0.98 & 0.98 & 0.99 & 0.93 & 1.0 & 0.0 & 0.99 & 0.89 & 0.99 & 0.86 & 0.98 \\
1.0 & 1.0 & 0.97 & 0.76 & 0.88 & 0.99 & 0.0 & 0.98 & 0.97 & 0.99 & 0.99 \\
0.97 & 0.97 & 0.93 & 0.89 & 0.99 & 0.89 & 0.98 & 0.0 & 0.92 & 0.79 & 0.97 \\
1.0 & 0.95 & 0.97 & 0.91 & 0.94 & 0.99 & 0.97 & 0.92 & 0.0 & 0.88 & 0.98 \\
0.99 & 0.97 & 0.86 & 0.93 & 0.99 & 0.86 & 0.99 & 0.79 & 0.88 & 0.0 & 0.77 \\
0.99 & 0.90 & 0.71 & 0.99 & 0.97 & 0.98 & 0.99 & 0.97 & 0.98 & 0.77 & 0.0
\end{bmatrix} \tag{3.3}$$

Figure 3.3: Example of dissimilarity matrix for Hoeffding Tree

The matrices calculated for each classifier are then passed to the modified DIANA technique which are explained in the subsection below.

3.2.2 Modified DIANA technique

The modified DIANA algorithm is very similar to the explained in the previous chapter. Truthfully, and as it was explained before, each classifier has its own Dissimilarity Matrix (which is the fact that makes it modified since the DIANA algorithm only uses one). To pick the first element in the splinter group, the element that disagrees the most with the others, it is calculated the Average Dissimilarity for each classifier. The classifier whose Average Dissimilarity is bigger will be the chosen one for that iteration of the DIANA technique. It is then, possible, to have different classifiers chosen for some branches of the tree.

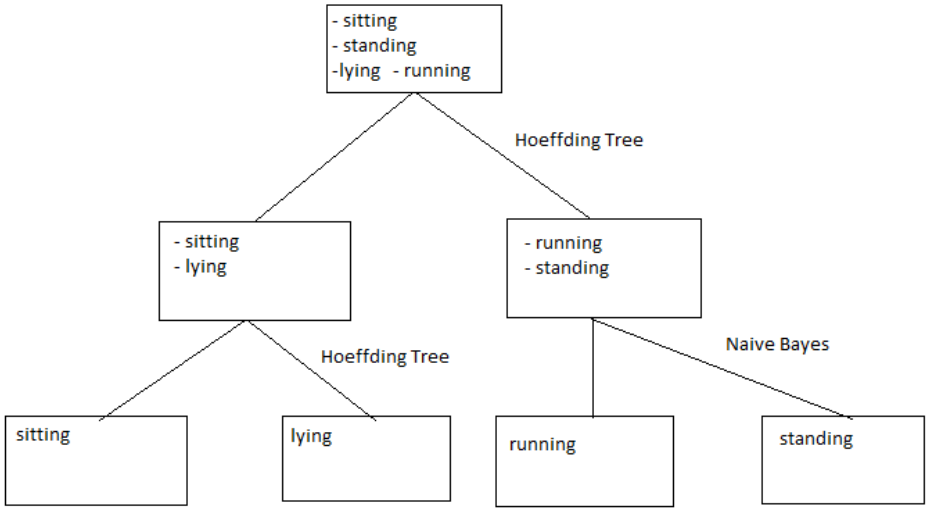

Figure 3.4: Example of how modified DIANA works

After the classifier and the splinter group has been initiated, the DIANA technique will resume and will keep iterating until a final state has been reached. When a node division is finished, if this cluster created is not a single cluster, this is, if the final cluster is constituted by only a final activity, the iteration for this cluster will cease, and will find nodes which are constituted by more than one activity so that they can be divided, and subsequently a classifier to be applied to this node division until a state where all existing clusters are only constituted by only one activity.

3.3 Tools

The tools we are going to use are very intertwined with the concepts of machine learning.

3.3.1 Massive *Online* Analysis

MOA is a data mining framework written in Java, developed by the University of Waikato, the same responsible for the development of WEKA. For our work, we must use with MOA since it is more directed to handle data streams which is exactly what we have to deal with. The purpose of this application is to provide an experimentation environment being able to supply classifiers, evaluation methods in such a way that the user can perform complex operations without much effort, using a graphical interface [BHKP10].

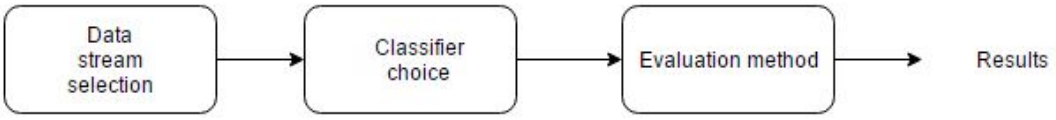

Figure 3.5: MOA workflow

In this project however, the API will used the most. This library in conjunction with existing code from INESC-TEC will provide a smoother environment for the development, since if it would be need to implement everything from root, the project would not be able to completed in a thesis span.

3.3.2 Android and Android Studio

In the beginning of the thesis it was supposed to port the results of the project and then implement a Android Application that could make a online prediction of the performed activity. As such, work went to the development of this application to extract data from sensors, such as microphone, gyroscope, among others. In the end, there was no time to port the results to the smartphone and the dataset to replace the current one was not collected, which is important since the results of the technique depend on the sensors used, and so the portability was not completed. Either way, the Android SDK was used in the project, therefore the mention.

3.3.3 Eclipse

To develop the application the Eclipse IDE was used. Eclipse also supports the development of JAVA which was an important requirement. Since the INESC-TEC project and the MOA API were prepared for the JAVA environment this language was always a default from the start.

3.4 Architecture

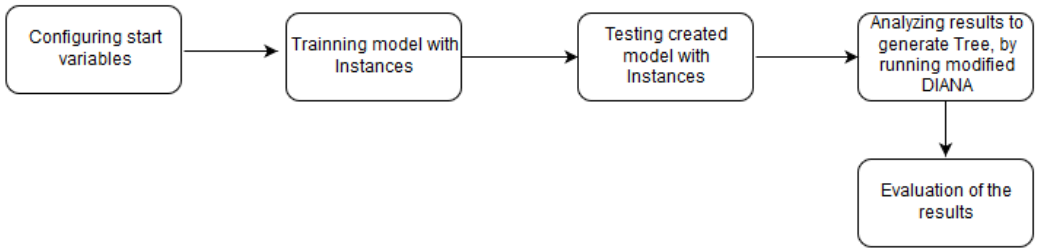

Figure 3.6: Architecture of the project

To explain this scheme in better detail, firstly it is necessary to input static arguments in the program to get the desired results. Variables such as overlapping, window size, classifiers to be used,

sensors. All of this will impact the tree created and consequently the final results. It is then necessary to train one model with instances while taking into account the start variables since the instances that will train the model will depend on them. After training the model, it is necessary to test it in order to obtain the results and create the confusion matrix and other necessary arguments to compute the modified DIANA technique. A tree will be generated with the data collected from the confusion matrix. When the tree is generated, all that is needed to do is test it to measure its accuracy Once this is done, it is essential to collect the results obtained and test them. This part will be more carefully explained in the sections below.

3.5 Dataset

The dataset [uci] used in this project came from Germany,it is publicly available, being that the main purpose of the dataset was to record 19 different activities: *sitting*, *standing*, *lying*, *walking*, *walking ascending stairs*, *descending stairs*, *running*, *cycling*, *nordic walking*, *watching TV*, *computer work*, *car driving*, *vacuum cleaning*, *ironing*, *folding laundry*, *house cleaning*, *playing soccer*, *rope jumping*. The activities were recorded by simply asking the users to perform the activities to be measured.

While it was supposed that this dataset would have all these activities, what happens in practice is that there were activities that were not present in the dataset at all so a preprocessing had to be done. As such the program was programmed in such a way to discard activities where there would be zero correct results(prediction/classification), since this value means the number of instances is too low to rely on the results.

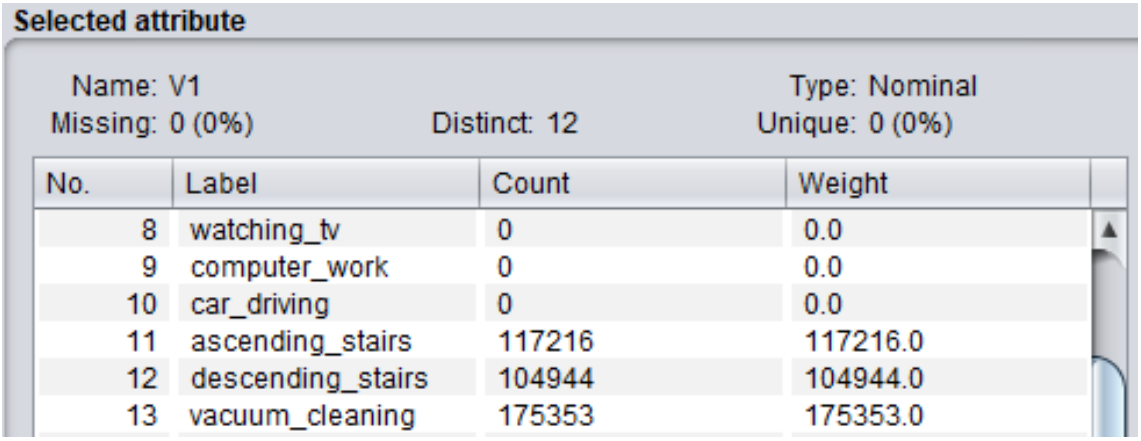

Figure 3.7: Number of Instances of some activities in PAMAP dataset

As it is visible in *watching tv*, *computer work*, and *car driving* there are zero existing instances, so this activity cannot be used. These activities were recorded at a 100 HZ sampling rate and it was used three IMU(Inertial measurement unit) in three different places(over the wrist on the dominant arm , on the chest, on the dominant side's ankle) as well as a heart rate monitor, with a sampling rate of 9 HZ. Because the API of MOA did not accept files the format that this dataset was at, this

demanded some preprocessing to convert it into ARFF, which the file extension which the API works with. Because of some memory and time overload, it was also necessary to divide the files into every person that participated on it.

3.6 Feature Extraction

In order to being able to analyze and proceed to create a model for training or testing, it is relevant to figure if the attributes that are being extracted from the sensors are appropriate or not. The selected features were part of the INESC-TEC project, so that part has already been studied and done. Features include mean as well as standard deviation alone and together, which means that both X,Y,Z get used alone and together in simultaneous. Because these attributes have already been studied, there is not a lot to say about this subject.

3.7 Semi Supervised Learning

The INESC TEC project which this study was built on, possesses a component of Semi Supervised Learning, meaning that it takes on unlabeled data to incorporate it to its training model. As such, the project it was built on also possesses this ability. To put it in simple terms, it basically means that, when testing instances, previous instances from the testing user will be taken into consideration, conducing to an increase in the overall accuracy.

3.8 Experiments

3.8.1 Experiments Setup

As seen in the section 3.4, the architecture, it is obvious that the setup of the variables fits in the first step presented on that section. Based on sections above, where it was explained some variables needed to be setup before running the project, it was necessary to research for the optimal point that could be achieved. In [Car16] it is mentioned that the optimal setup was 200 HZ and 70% overlap. The overlap means to evaluate a current window, it will receive 70% of the previous window to make a decision. However, after some testing, it turned out that the optimal window was around 400. The datasets consists of the dataset of eight people, in spite of the fact there needs to be unused instances for testing, and in the case of this particular project it is needed two groups of testing: one for creating the tree and other one for testing the created tree. It was used the three classifiers described in the previous chapter: KNN, Naive Bayes and Hoeffding Tree. Firstly, it was just used the ankle sensors. For the second experiment, all motion sensors were enabled. The reason for this was to study how more or less data/selected features would affect the results.

3.8.2 First Experiment

Using the technique described in the sections above, the parameters mentioned in the subsection and the preprocessing of the activities, it possible to retrieve a tree.

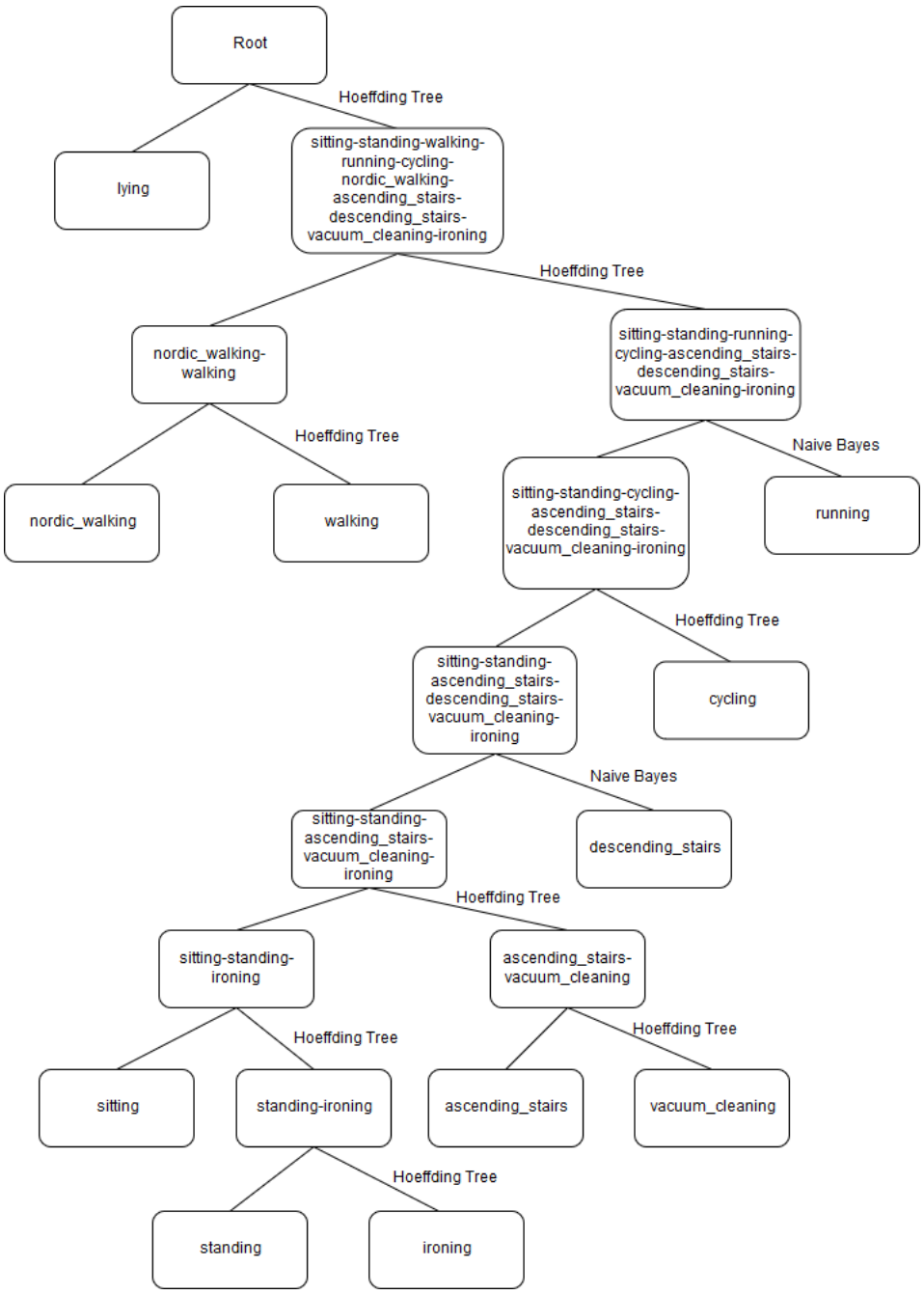

Figure 3.8: Tree generated by running project setup

At each point of decision of the tree, there is a classifier associated with it, which was the used classifier for that specific division. The tree in itself is pretty self explanatory, so now the results will be presented.

3.8.2.1 Training and testing of models

In order to obtain the results in the generated tree, it is necessary to iteratively train and test multiple models. For each time, a classifier is required to use in order to divide a branch of the tree, a new model is trained and tested. The results are collected, and in case the next node is a not simple node(that is, a node with a simple activity, in which case there is no more further division), a new model is trained with the activities regarding that node, and with the instances that were classified from the node above the new model is tested. When a leaf node is reached, the iteration stops.

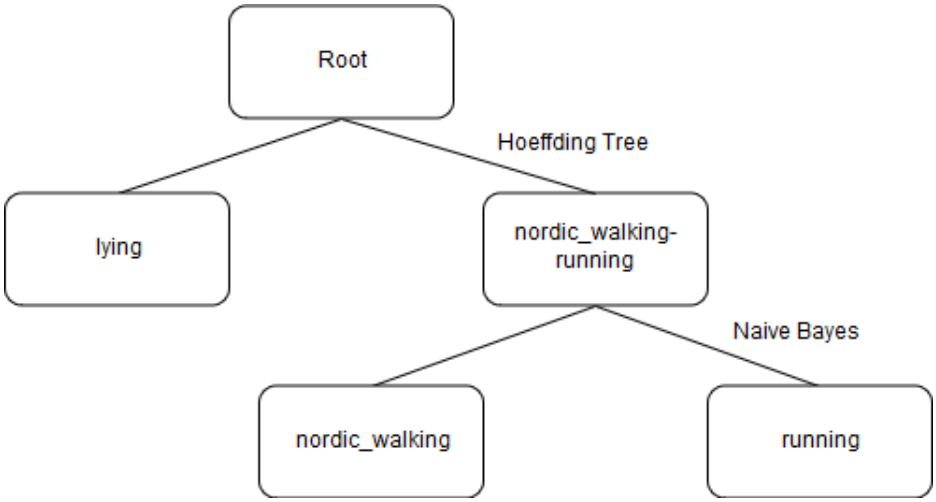

Figure 3.9: Tree example

So, imagining that this is the tree we are trying to test, there are several steps to be done in order to obtain the accuracy of the tree. First, a model for the root node is trained. Since the root node consists of *lying*, *nordic_walking* and *running*, this training will be constituted by two groups: one group constituted by the *lying* instances and the other one consisting of both *nordic_walking* and *running*. Therefore, once the training is done and the testing phase begins, an instance will be either classified as *lying* or *nordic_walking-running*. Once the testing for this node is over, there will be some instances classified as *lying* and some classified as *nordic_walking-running*. Because the *lying* node is a leaf node, since it is only constituted by one activity, it cannot be divided any more and thus, it is easy to see which instances were classified as *lying* versus all *lying* instances and create an accuracy measure for this tree, for this activity. However, there is still nodes in the tree which are not leaf nodes, and that were not explored. It is now necessary, to continue testing and collecting results to train the node *nordic_walking-running* with both instances of *running* and *nordic_walking*, but this time, representing separate groups, while in the previous one they were grouped together. Using the specified classifier, Naive Bayes in this case, a prediction is made and since both children of *nordic_walking-running* are leaf nodes the iteration stops. Now all the starting testing instances are assigned a leaf node, and therefore all activities accuracy can be calculated and measured. This of course, only represents the process for one tree, which

corresponds to one user. In order to obtain accurate results, the evaluation of the results will make use of multiple techniques described in the section below.

3.8.3 Second Experiment

The setup used for this experiment was exactly the same as the first one with one particularity. Instead of using only the ankle sensors, all the sensors available in PAMAP dataset were used. Therefore the generated tree was the following:

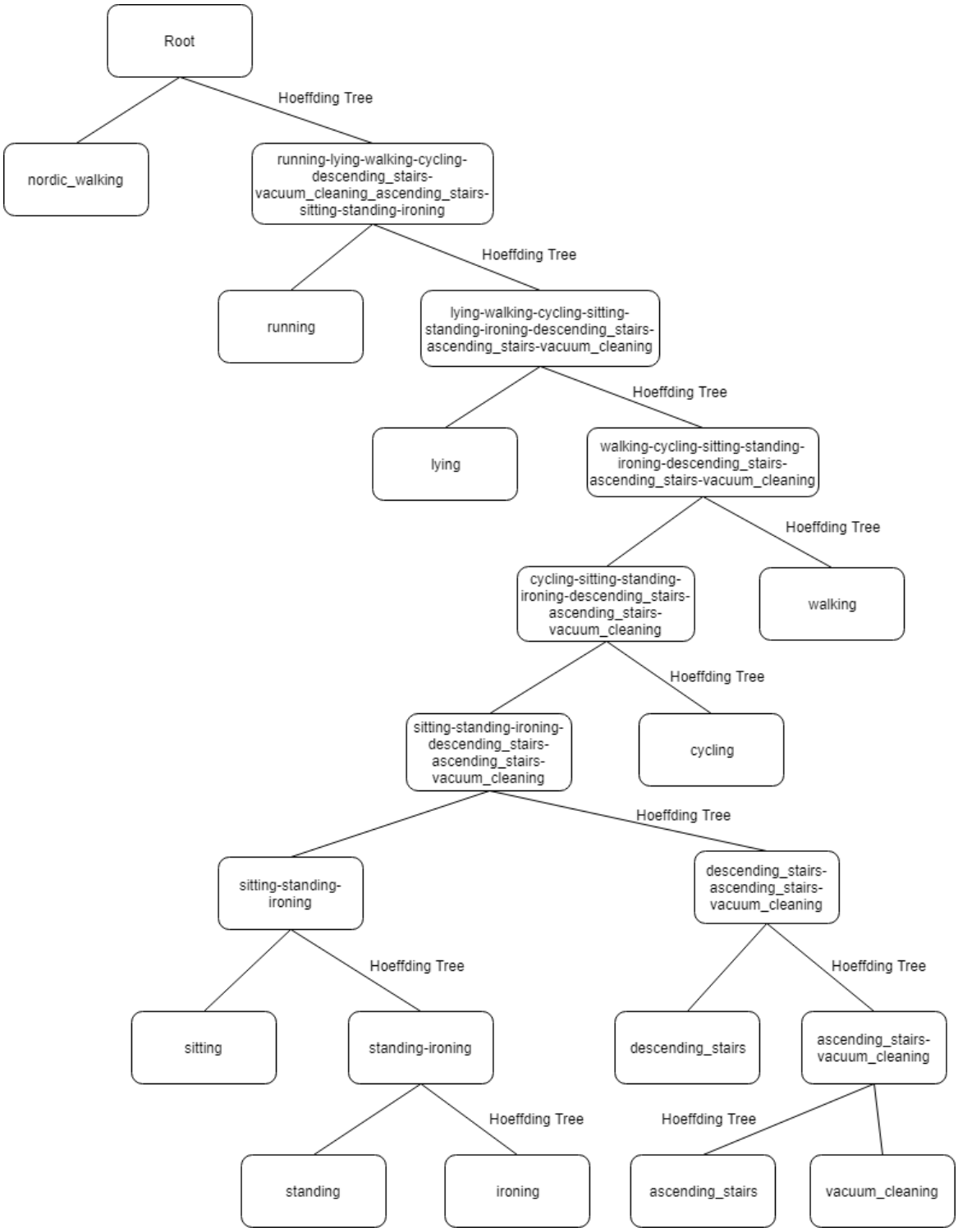

Figure 3.10: Tree generated with data from all sensors

3.9 Evaluation of results

In previous chapters, the holdout, as well as the leave one out methods were mentioned. These were the techniques used in this study in order to mitigate non-accurate results. In order to run the DIANA algorithm, it is needed some samples to generate the tree(in this study, a whole user is left for this). As such, a random user that possessed almost all activities was picked for this task, in this case, the user 6. This part of the evaluation of the results is, therefore, the holdout method. However, this was only for the generation of the tree. To obtain data regarding on how well this tree classifies new instances, another user has to be made available to the testing of the generated tree. This time, a leave one out approach is used, where the user used for testing the tree is rotated between all the users(except the user 6, since that one is fixed) and, in the end, the results are an average of the results collected between all users.

Proposed Method

Chapter 4

Presentation of results and discussion

The collection of the results used the Holdout as well as the leave-one-out evaluation tactics, since, to generate the tree, it is necessary one user to calculate the confusion matrices that allow the usage of the modified DIANA algorithm. The user picked to create the tree is the user number 6. This represents the Holdout evaluation part. But, it is still necessary to test how well the generated tree behaves. To do so, the leave-one-out evaluation, and the user that is left out is rotated, and the instances are summed up, both the correct and the total. As such, the results presented below are the result of all the trees generated tested with a different user.

Activity	**Total Instances**	**Correct Instances**	**Accuracy**
Lying	1414	949	67.1%
Nordic Walking	1346	626	46.5%
Walking	1773	443	25.0%
Running	610	343	56.2%
Cycling	1199	282	23.5%
Descending Stairs	779	417	53.5%
Ascending Stairs	868	468	54.0%
Vacuum Cleaning	1284	435	33.9%
Ironing	1674	889	53.1%
Sitting	1351	329	24.4%
Standing	1380	876	63.5%

Table 4.1: Results of the first experiment

The Naive Bayes had an average accuracy of 68.82%. The Hoeffding Tree had an average accuracy of 71.1%. The KNN classificar had an average accuracy of 14.1%. The average accuracy of the implemented technique is of 45.6%. The data presented here was the average for all users.

4.0.0.1 Collection of results for second experiment

The collection of the results use the same setup and tactics(holdout, leave one out) of the first experiment.

Activity	Total Instances	Correct Instances	Accuracy
Lying	1414	937	66.3%
Nordic Walking	1346	582	43.2%
Walking	1773	1574	88.8%
Running	610	474	77.7%
Cycling	1199	818	68.2%
Descending Stairs	779	542	69.6%
Ascending Stairs	868	546	62.9%
Vacuum Cleaning	1284	681	53.0%
Ironing	1674	1393	83.2%
Sitting	1351	314	23.2%
Standing	1380	1006	72.9%

Table 4.2: Results of the second experiment

For the same setup, it is important to note that both Naive Bayes and Hoeffding Tree achieved 85.1% accuracy for all activities, which is significantly better than the results achieved by the hierarchical approach, that barely got one activity over this accuracy level (walking - 88.8%), and across all the activities the average was 64.5%. The KNN classifier achieved a mere 35.45%, because of the way KNN works when making a prediction. It sees the most close neighbors to the instance that is being classified and according to those it makes a preponderant decision. The best it would be that in each prediction every instance would be taken into consideration, since it would mean that when making a prediction every instance would be taken into consideration, however, due to computational constrictions, a more real number needs to be defined. To investigate now how the approach matches in each activity, an average for all the users for all the activities is made and showed in the table below. The KNN accuracy was not shown due to the very low accuracy presented by the classifier.The Hoeffding Tree/Naive Bayes showed the same matrices throughout the users so the accuracy is the same for both classifiers.

Activity	Accuracy Hierarchy(DIANA)	Accuracy Hoeff/Naive Bayes
Lying	66.3%	92.2%
Nordic Walking	43.2%	48.5%
Walking	88.8%	75.2%
Running	77.7%	90.1%
Cycling	68.2%	85.1%
Descending Stairs	69.6%	74.9%
Ascending Stairs	62.9%	86.3%
Vacuum Cleaning	53%	91.5%
Ironing	83.2%	80.5%
Sitting	23.2%	69.2%
Standing	72.9%	55.3%

Table 4.3: Comparison between Hierarchical approach and normal classifiers(average of all users)

The multi classifier approach outperformed the hierarchical one(DIANA). There are however, some activities that actually outperform the first approach like *walking,ironing,standing*, however it is not enough to compensate the failures on detecting such activities like *lying,ascending stairs,vacuum cleaning, sitting*. The multi-class approach proves to be superior versus the implemented approach.

4.1 Interpretation of Results

There are some key factors necessary that can utterly explain some of the results obtained, and that will be discussed below. While some may affect both the multi-class classification and the hierarchical classification, it is still important to not try to justify differences between one another, but also explain they both had the accuracy that they did.

4.1.1 Calibration of sensors

This is error of the hardware that captures the data from the sensors. If the sensors are not well calibrated errors can be produced. Because this dataset is widely used, it is assumed that it is correctly setup, however we cannot be 100% sure of this.

4.1.2 Propagation of errors

Nodes that are located on the superior part of the tree are less prone to errors. In fact, if an instance has to be classified seven times before reaching its respective leaf node while another instance only needs to be classified once, this instance will have a lot higher accuracy than the ones that needs to be classified seven times, since in these classifications some instances can be lost mid-way to another activities, while the activities in topper levels do not suffer from this problem. This

problem, mentioned in [BDP+13] can begin to explain why activities like *sitting, cycling* never present a really high accuracy on any of the users.

4.1.3 Dissimilarities between nodes

It is essential to remind that the tree is based on the dissimilarities between activities shown in the distance matrices. And, as such, the tree is grouped according to these results. However, not all the times, the distance between nodes is enough to get a clear distinction. What this means is that, while it is relatively easy to decide if an instance is either *lying* or *walking-running-ascending_stairs-ironing*, it is not that easy to differentiate if an instance belongs to *walking* or *nordic_walking*. This was an observed problem where instances would have high accuracy until the final node, just to be confused one for the other, lowering the accuracy of both significantly. Specially with similar activities, like the previously mentioned, it was observed in a particular node *nordic_walking-walking* to have a high percentage of the instances actually being a part of the node. However, when the time to divide *nordic_walking* and *walking* came, half of the instances belonging to *nordic_walking* would be classified as *walking* and vice versa(when using the ankle sensors only). In the experiment with all the sensors however, *walking* was one of the activity with most accuracy. The dissimilarities between the nodes are deeply connected with the selected features. Those are used by the modified DIANA algorithm in the classification of the instances in order to get results to produce the tree. However, if the selected features are not appropriate or not in enough number, it will impossible to distinguish two activities, making it impossible to distinguish two given nodes. To put it in simple terms, sometimes either the division made by the DIANA algorithm is not the most efficient, because of poorly selected features that can distinguish two nodes, making it impossible to recognize any dissimilarities that can be used.

4.1.4 The dataset

The PAMAP dataset consists of eight users. A very low number on itself. Not only that, if a look is taken across the generated trees for each user, it should not happen that if a user used for testing suddenly begins being used for training and another one is picked for testing, that the tree changes. The data should be enough that even when tested with other users, that it stays the same (experiment with ankle sensor only, since when added all sensors the tree started to be more constant).

4.1.5 Over-fitting

It is known that decision trees(Hoeffding Tree) and Naive Bayes are prone to over-fitting. Over-fitting is present also in the results. It is visible because there are some activities that have very good accuracy on a lot of users, but suddenly, there is a few group of users that has a percentage of close to 0% for the activity. Considering the ankle sensor experiment only, for most users *lying* has even more than 90% accuracy. However, for two of them, the accuracy is around 1%. Considering this, it would be easy to assume that the instances suffered from point number one

which was the propagation of errors. Although, it is safe to exclude propagation of errors since in both cases *lying* is an activity located on the top part of the tree. And it cannot be also because of the selected features since most of the other users were able to detect the activity with such a big accuracy. Therefore it must be because the users performed the activity *lying* in a way the others did not(along with probably some noise). To conclude, it is important to note that while most of them affect both the multi-class classification and the hierarchical classification, the latter is still affected by the propagation of errors that the multi-class classification does not suffer from. Every other problem, it is shared, which might explain why the multi-class classification approach was more successful.

4.2 The tree

There are some good points presented by this approach however.For the ankle experiment, intuitively, there is a certain degree of logic present on the tree. *nordic_walking* and *walking* are grouped together, and that makes sense since they are both types of walking. *Lying* is also different from the other activities, and it is represented in the tree as the first division. Even *descending_stairs* and *ascending_stairs* are together as a node until a point. It is also noticeable that most of the locomotion related activities regarding the ankle activity(*nordic_walking*, *walking*, *cycling*, *running*) are located on the top side of the tree(on both experiments), while the activities that do not benefit with this sensor as much are located on the bottom part of the tree.

4.3 Classifiers accuracy vs Tree Accuracy

Every classifier used, except for KNN has an higher accuracy than the technique created. The reason why the KNN has such a low accuracy is because, in order to be faster when classifying it is requires to set a number of instances N, that the classifier will take into account when making a prediction. If the classifier takes into account all the instances, it will be very slow. In order to speed up the process a N more reasonable is picked. As such, the classifier does a prediction based on only some instances which leads to a low accuracy. As for the rest of the classifiers, they both do pretty well compared to the newly implemented approach. It is however necessary to note the power that this approach can also have. On some users, almost all the activities reach a very high accuracy. To give a concrete example, *lying* has an accuracy of around 90% to most of users. However, in others it goes down to 0% accuracy. Because there are a lot of users that do possess 90% accuracy it cannot be that the selected features are not enough. It cannot be blamed on propagation of errors as well, since in most trees, this activity is located on the top part. What happens here is that *lying* of some users is badly recognized as other activities, which signals over-fitting meaning that the classifier can only get a perception of what lying should be like, in terms of data received, for some users. This problem is also present in the multi-classifier approach where some users also have an activity with over 90% accuracy, and on some it goes back to around 0%. Despite being partially due to over-fitting, it is also due to lack of data.

Chapter 5

Conclusion

The purpose of the work developed was to study hierarchical classification by means of hierarchical clustering in the context of human activity recognition. Despite the results being disappointing, it still sheds an important light on what works and what does not in order to improve in the future. Human Activity Recognition is a vast field of studies with a lot of potential to grow, mostly due to its complexity and a the number of variables to study, while trying to keep the approach computationally low or, in case of a portable device, low energy consumption. Still, there are a lot of works done in spite of this complexity, with interesting approaches and promising results that with deeper research can improve quality of life of thousands of lives and even prevent death all around the globe.

5.1 Future Work

From this study multiple ideas can and should be explored. This approach studied binary hierarchical classification with hierarchical clustering. The next logical step, and because it was proved in this work that multi-classifier approaches can also be powerful, would be the integration of both, creating a tree with multiple classifiers and multiple classes.

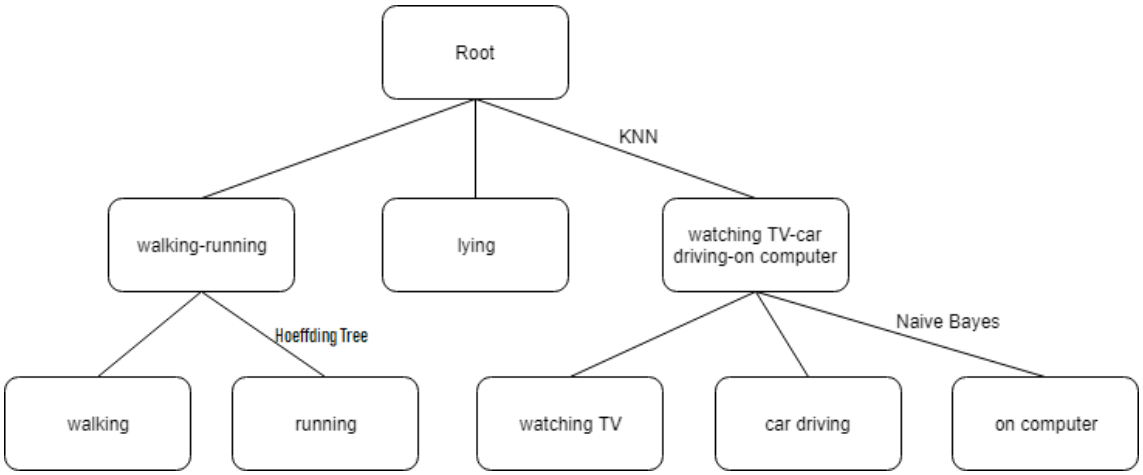

Figure 5.1: Example of described approach

Another concept that could drastically improve the performance of this approach would be the introduction of context to the tree. Explaining in more detail, the system should be able to detect flaws as it incorporates the context of the activity. Explaining this in greater example, if the user is *running* it is expected that at some point there will be *walking*. Or, imagining the system detects the user is outdoors is not likely that he is *washing teeth*. This approach, would lead to a further increase in accuracy. Nonetheless, Human Activity Recognition is a vast field, as said before, therefore there is an immense number of viable options, new sensors or selected features to be studied that can lead to a very successful approach.

References

[BBA+16] Henrik Blunck, Sourav Bhattacharya, Allan Stisen and Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, and Mads Møller Jensen and Tobias Sonne. Activity recognition on smart devices: Dealing with diversity in the wild. *GetMobile*, 20(1):34–38, January 2016.

[BDP+13] Oresti Banos, Miguel Damas, Hector Pomares, Fernando Rojas, Blanca Delgado-Marquez, and Olga Valenzuela. Human activity recognition based on a sensor weighting hierarchical classifier. *Soft Computing*, 17(2):333–343, 2013.

[BHKP10] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[Car16] Hugo Louro Cardoso. Predicting activities from smartphones. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2016.

[CWZ+17] Liang Cao, Yufeng Wang, Bo Zhang, Qun Jin, and Athanasios V Vasilakos. Gchar: An efficient group-based context—aware human activity recognition on smartphone. *Journal of Parallel and Distributed Computing*, 2017.

[dOL12] Alexandre de Oliveira Lopes. Activity recognition from smartphone sensing data. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2012.

[dS13] Joana Raquel Cerqueira da Silva. Smartphone based human activity prediction. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2013.

[EPMK08] Miikka Ermes, Juha Pärkkä, Jani Mäntyjärvi, and Ilkka Korhonen. Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. *IEEE transactions on information technology in biomedicine*, 12(1):20–26, 2008.

[FDFC10] Davide Figo, Pedro C. Diniz, Diogo R. Ferreira, and João M. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal Ubiquitous Comput.*, 14(7):645–662, October 2010.

[KR09] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[LL13] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials*, 15(3):1192–1209, 2013.

[LY13] Miguel A Labrador and Oscar D Lara Yejas. *Human Activity Recognition: Using Wearable Sensors and Smartphones*. CRC Press, 2013.

REFERENCES

[MRS12] Corey McCall, Kishore K Reddy, and Mubarak Shah. Macro-class selection for hierarchical k-nn classification of inertial sensor data. In *PECCS*, pages 106–114, 2012.

[MSSD06] Uwe Maurer, Asim Smailagic, Daniel P Siewiorek, and Michael Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pages 4–pp. IEEE, 2006.

[Nai] Naive bayes formula. http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/image/bayesian/bayestheorem.jpg.

[NJ02] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[QMXW10] Huimin Qian, Yaobin Mao, Wenbo Xiang, and Zhiquan Wang. Recognition of human activities using svm multi-class classifier. *Pattern Recognition Letters*, 31(2):100 – 111, 2010.

[uci] Uci machine learning repository: Pamap2 physical activity monitoring data set. http://archive.ics.uci.edu/ml/datasets/pamap2physicalactivitymonitoring.

[WLA+09] Yi Wang, Jialiu Lin, Murali Annavaram, Quinn A. Jacobson, Jason Hong, Bhaskar Krishnamachari, and Norman Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, MobiSys '09, pages 179–192, New York, NY, USA, 2009. ACM.

[XHZ+18] Cheng Xu, Jie He, Xiaotong Zhang, Haipiao Cai, Shihong Duan, Po-Hsuan Tseng, and Chong Li. Recurrent transformation of prior knowledge based model for human motion recognition. *Computational Intelligence and Neuroscience*, 2018, 2018.

[XMZT17] Wanru Xu, Zhenjiang Miao, Xiao-Ping Zhang, and Yi Tian. Learning a hierarchical spatio-temporal model for human activity recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 1607–1611. IEEE, 2017.

[Zhe15] Yuhuang Zheng. Human activity recognition based on the hierarchical feature selection and classification framework. *Journal of Electrical and Computer Engineering*, 2015:34, 2015.

Appendix A

Results

A.0.1 Results for first experience

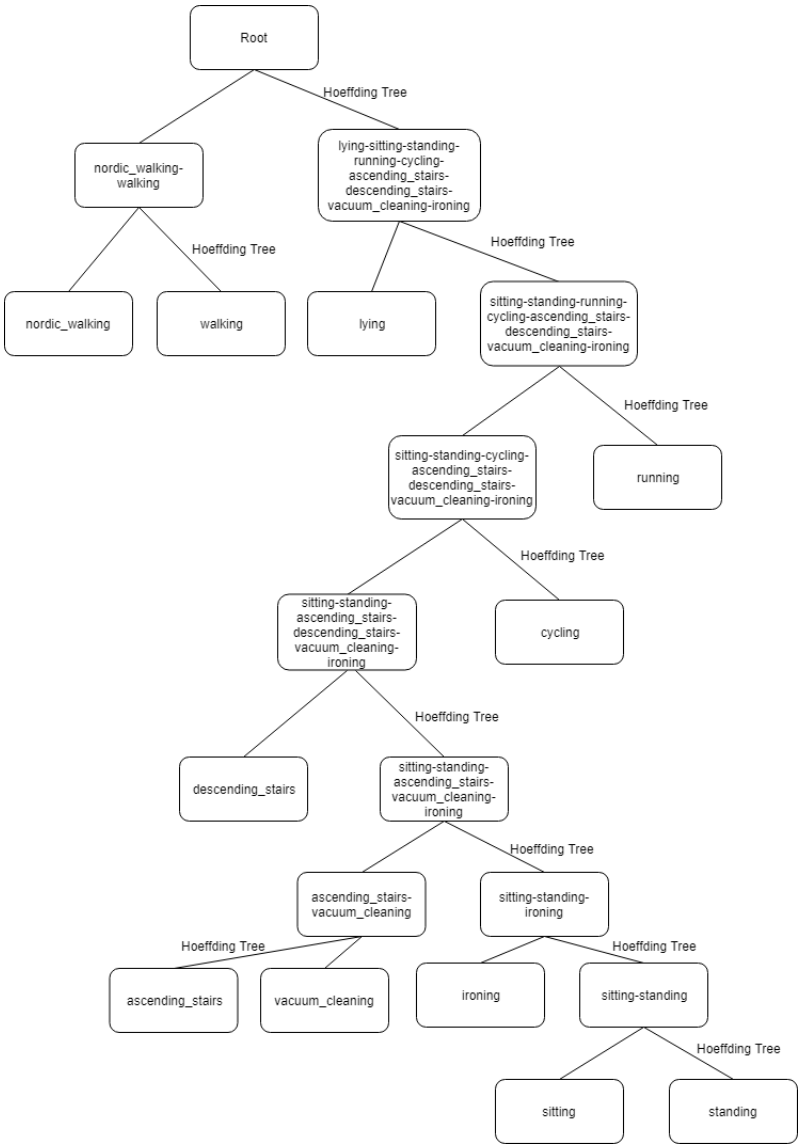

Figure A.1: Tree generated by testing with user 1

Activity	**Total Instances**	**User 1**	**Accuracy**
Lying	227	209	92.1%
Nordic Walking	169	18	10.7%
Walking	185	30	16.2%
Running	174	169	97.1%
Cycling	197	80	40.1%
Descending Stairs	124	39	31.4%
Ascending Stairs	133	69	51.9%
Vacuum Cleaning	191	25	13.1%
Ironing	196	82	41.8%
Sitting	196	0	0%
Standing	181	90	49.7%

Table A.1: Results by testing the tree with user 1

User 1: KNN - 147(7.06%) Naive Bayes - 1473(70.78%), Hoeffding Tree - 1473(70.78%)

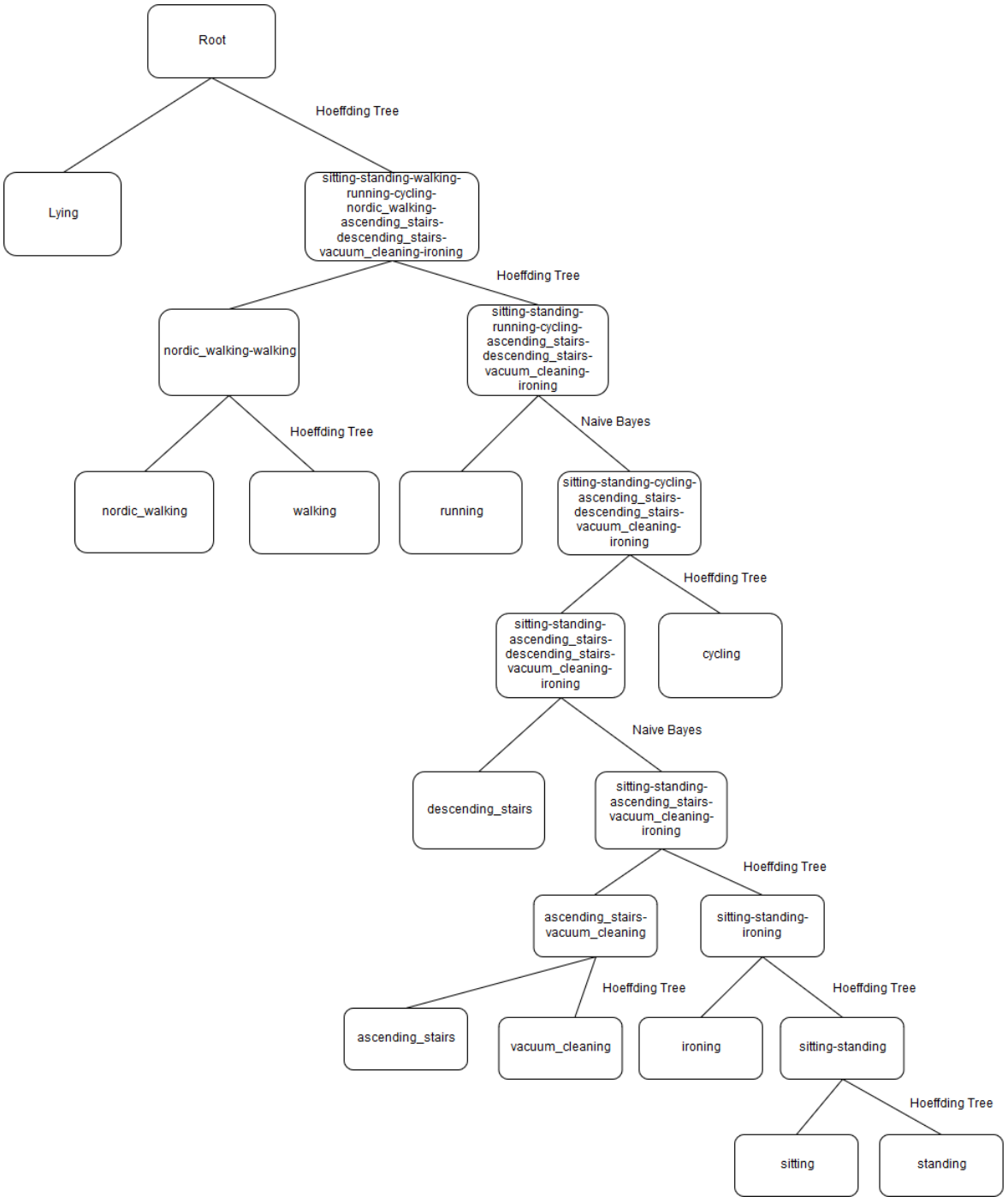

Figure A.2: Tree generated by testing with user 2

User 2: KNN - 151(7.32%) Naive Bayes - 1371(65.88%), Hoeffding Tree - 1492(71.70%)

Activity	**Total Instances**	**User 2**	**Accuracy**
Lying	196	181	92.3%
Nordic Walking	247	171	69.2%
Walking	272	76	27.9%
Running	73	59	80.8%
Cycling	210	29	13.8%
Descending Stairs	126	80	63.5%
Ascending Stairs	145	130	89.7%
Vacuum Cleaning	172	94	54.7%
Ironing	241	57	23.7%
Sitting	186	21	11.3%
Standing	213	210	98.6%

Table A.2: Results by testing the tree with user 2

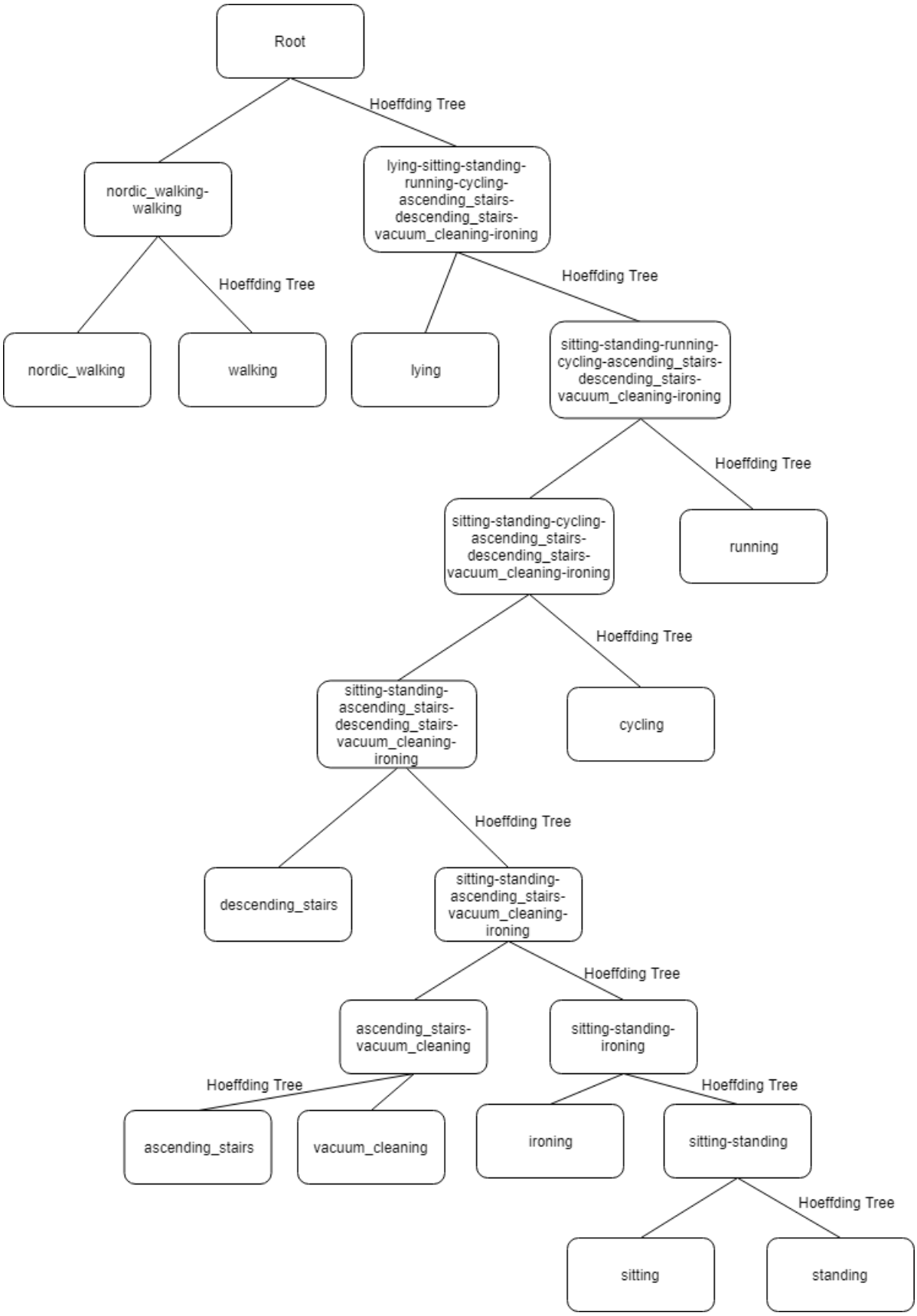

Figure A.3: Tree generated by testing with user 3

Activity	Total Instances	User 3	Accuracy
Lying	184	2	1%
Nordic Walking	0	0	Not Count
Walking	239	27	11.3%
Running	0	0	Not Count
Cycling	0	0	Not Count
Descending Stairs	127	79	62.2%
Ascending Stairs	87	62	71.3%
Vacuum Cleaning	169	90	53.3%
Ironing	233	153	65.7%
Sitting	240	0	0%
Standing	171	105	61.4%

Table A.3: Results by testing the tree with user 3

User 3: KNN - 174(8.36%) Naive Bayes - 1479(71.07%), Hoeffding Tree - 1537(73.86%)

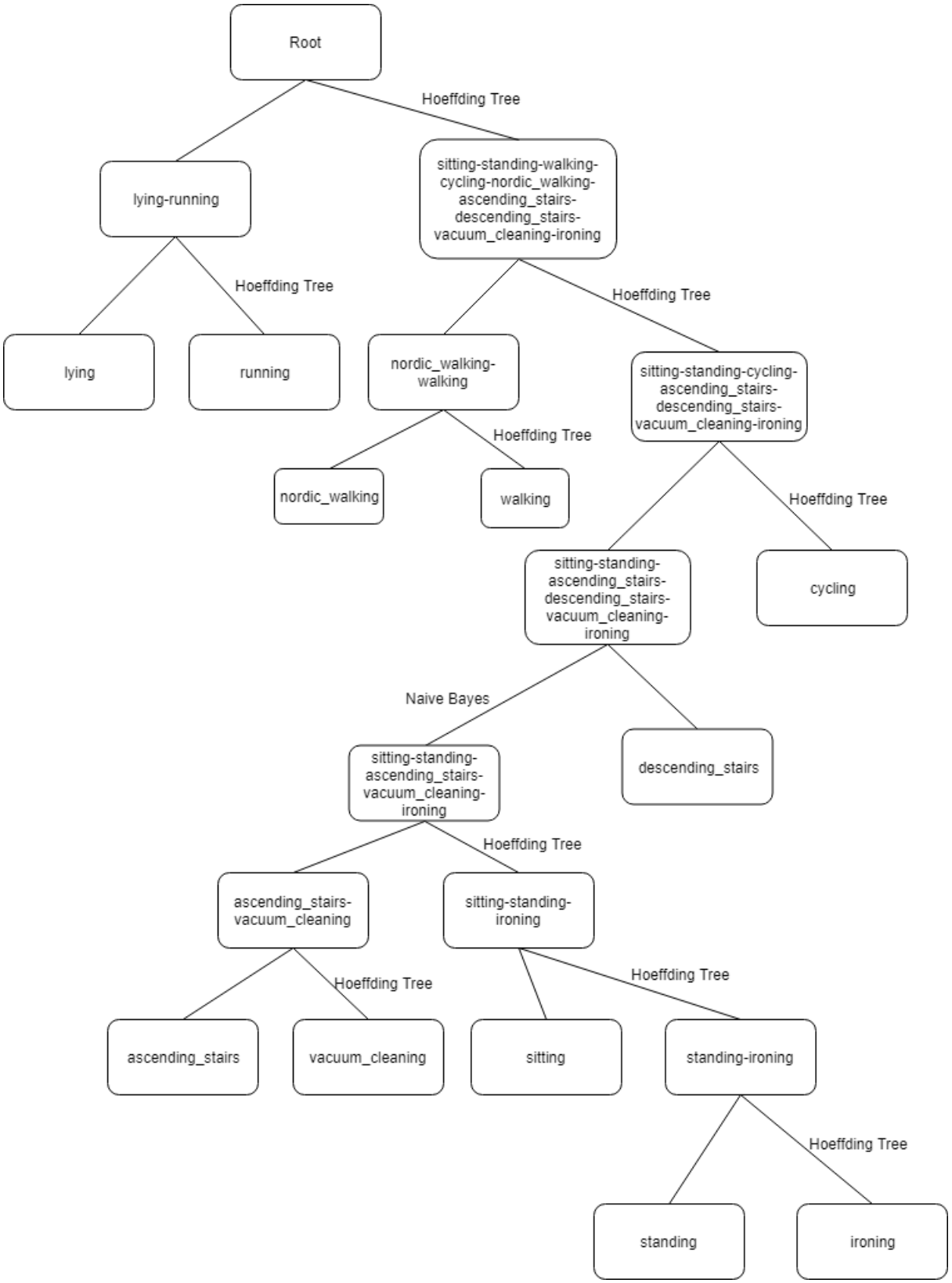

Figure A.4: Tree generated by testing with user 4

Activity	**Total Instances**	**User 4**	**Accuracy**
Lying	193	187	96.9%
Nordic Walking	230	124	53.9%
Walking	266	85	32.0%
Running	0	0	Not Count
Cycling	186	89	47.8%
Descending Stairs	119	38	31.9%
Ascending Stairs	139	108	77.7%
Vacuum Cleaning	167	60	35.9%
Ironing	208	191	91.8%
Sitting	212	154	72.6%
Standing	206	80	38.8%

Table A.4: Results by testing the tree with user 4

User 4: KNN-146(7.02%) Naive Bayes- 1416(68.04%) Hoeffding Tree-1444(69.39%)

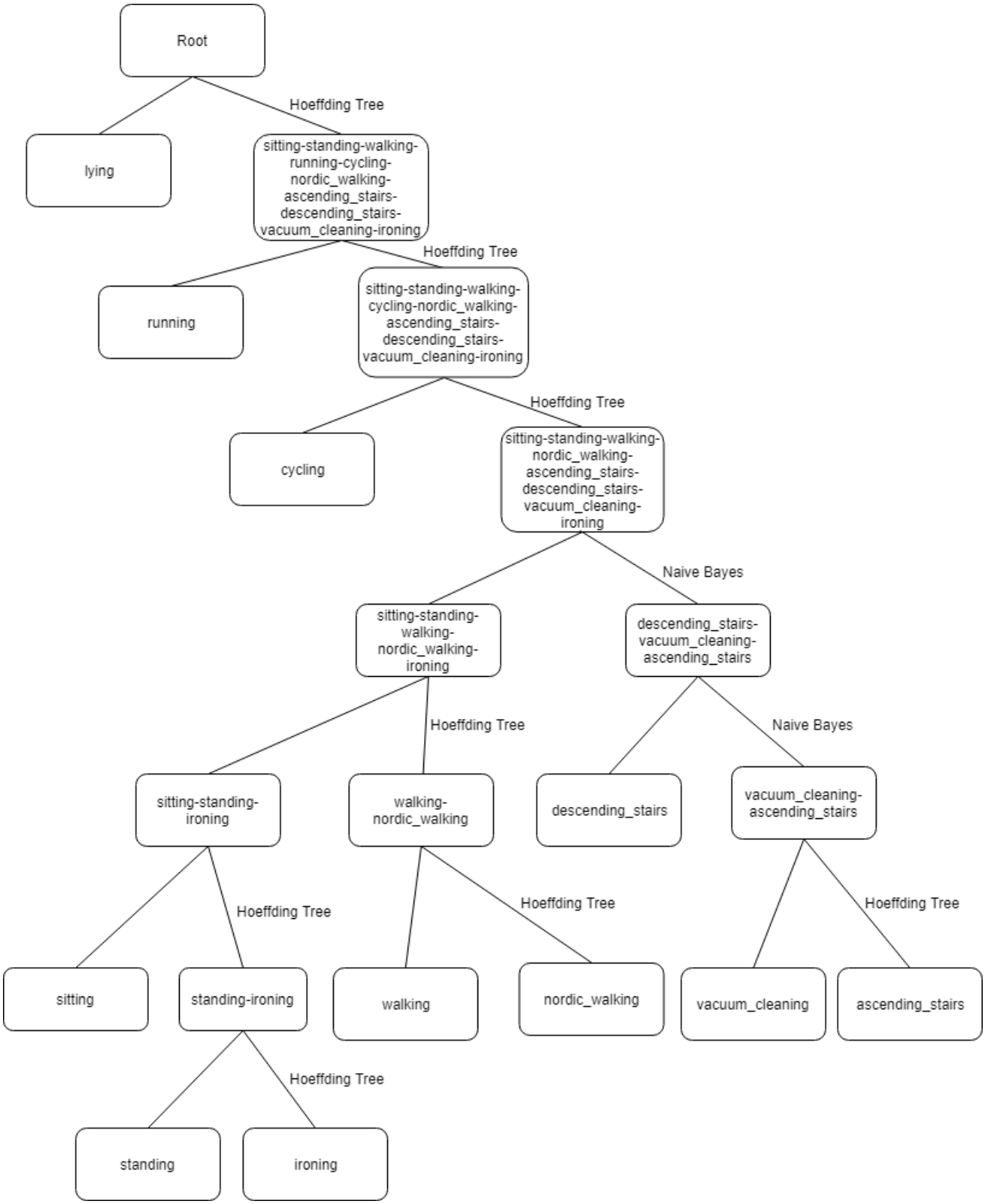

Figure A.5: Tree generated by testing with user 5

User 5: KNN - 163(7.83%), Naive Bayes 1363(65.50%), Hoeffding Tree 1440(69.20%)

Activity	**Total Instances**	**User 5**	**Accuracy**
Lying	198	182	91.9%
Nordic Walking	219	95	43.4%
Walking	267	113	42.3%
Running	202	34	16.8%
Cycling	205	55	26.8%
Descending Stairs	106	56	52.8%
Ascending Stairs	119	76	63.9%
Vacuum Cleaning	203	56	27.6%
Ironing	276	176	63.8%
Sitting	224	154	68.8%
Standing	184	87	47.3%

Table A.5: Results by testing the tree with user 5

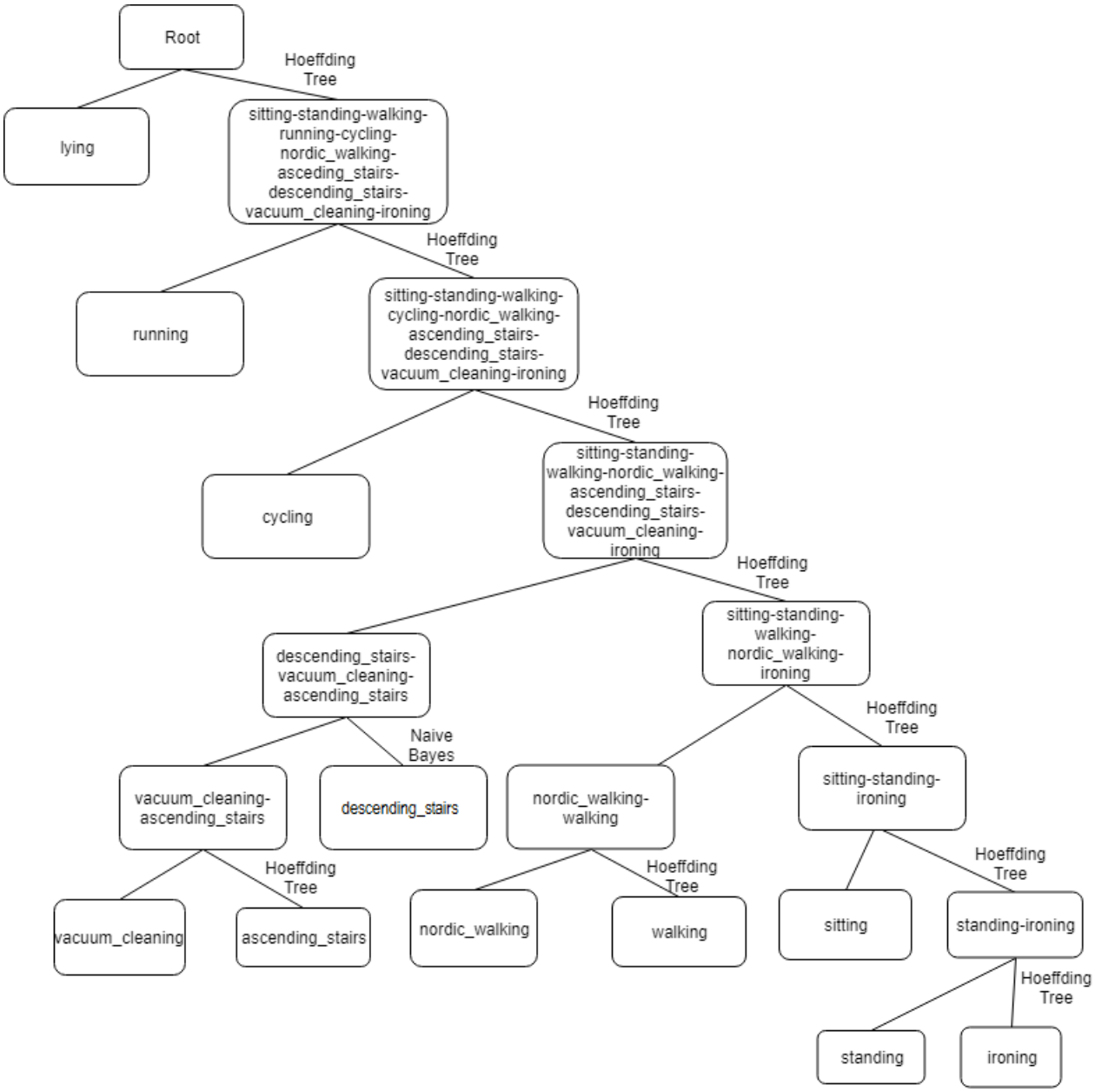

Figure A.6: Tree generated by testing with user 7

User 7: KNN - 161(7.74%), Naive Bayes 1400(67.28%), Hoeffding Tree 1467(70.49%)

Activity	**Total Instances**	**User 7**	**Accuracy**
Lying	214	0	0%
Nordic Walking	240	189	78.8%
Walking	281	34	12.1%
Running	27	22	81.5%
Cycling	189	28	9.5%
Descending Stairs	97	71	73.2%
Ascending Stairs	147	15	10.2%
Vacuum Cleaning	179	75	41.9%
Ironing	246	229	93.1%
Sitting	102	0	0%
Standing	215	115	53.5%

Table A.6: Results by testing the tree with user 7

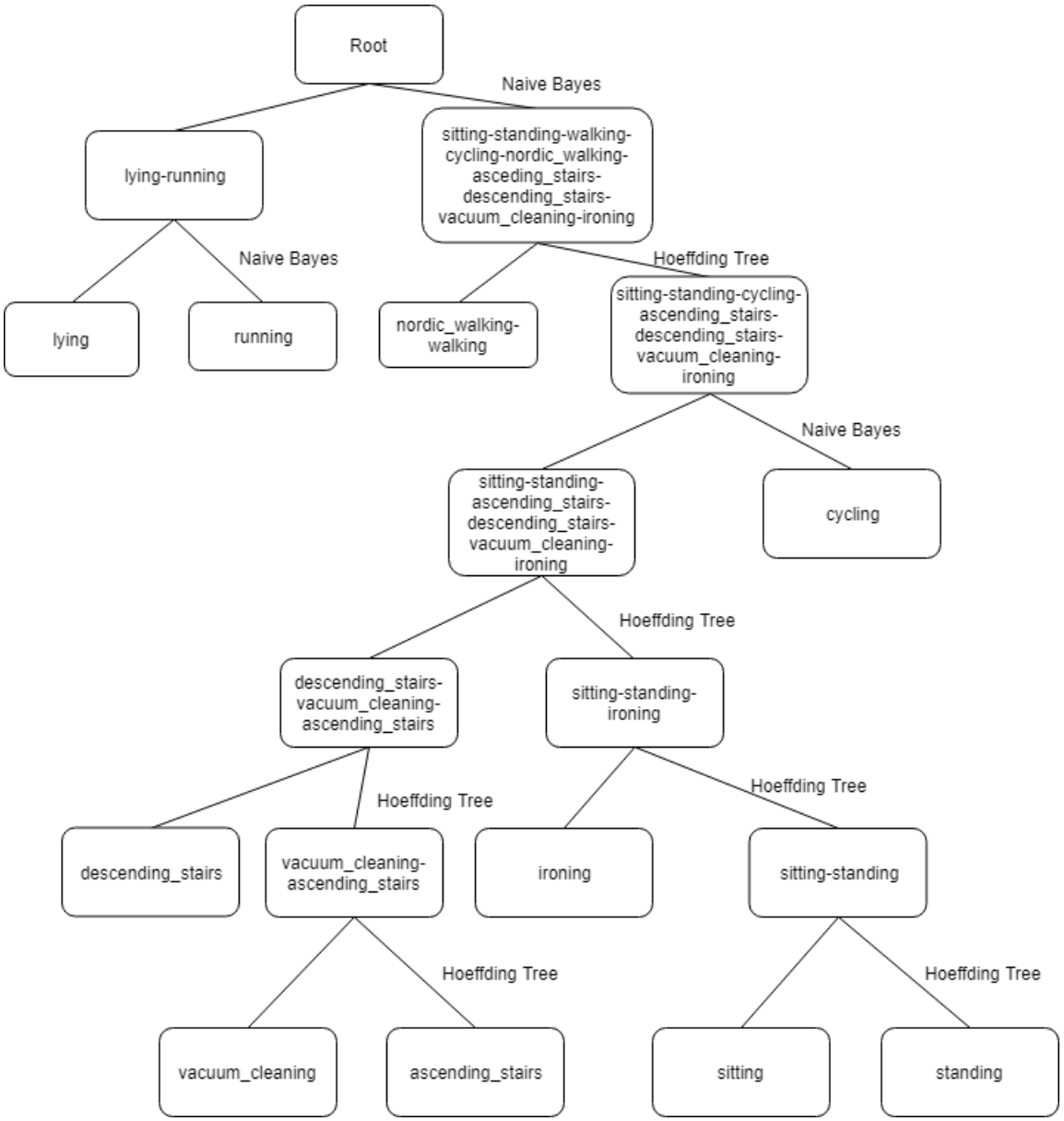

Figure A.7: Tree generated by testing with user 8

User 8 - KNN 1108(53.24%), Naive Bayes 1523(73.19%), Hoeffding Tree 1502(72.18%)

Activity	**Total Instances**	**User 8**	**Accuracy**
Lying	202	188	93.1%
Nordic Walking	241	29	12.0%
Walking	263	78	29.7%
Running	134	59	44.0%
Cycling	212	1	0.5%
Descending Stairs	80	54	67.5%
Ascending Stairs	98	8	8.2%
Vacuum Cleaning	203	35	17.2%
Ironing	274	1	0.4%
Sitting	191	0	0%
Standing	210	189	90.0%

Table A.7: Results by testing the tree with user 8

A.0.2 Results for second experience

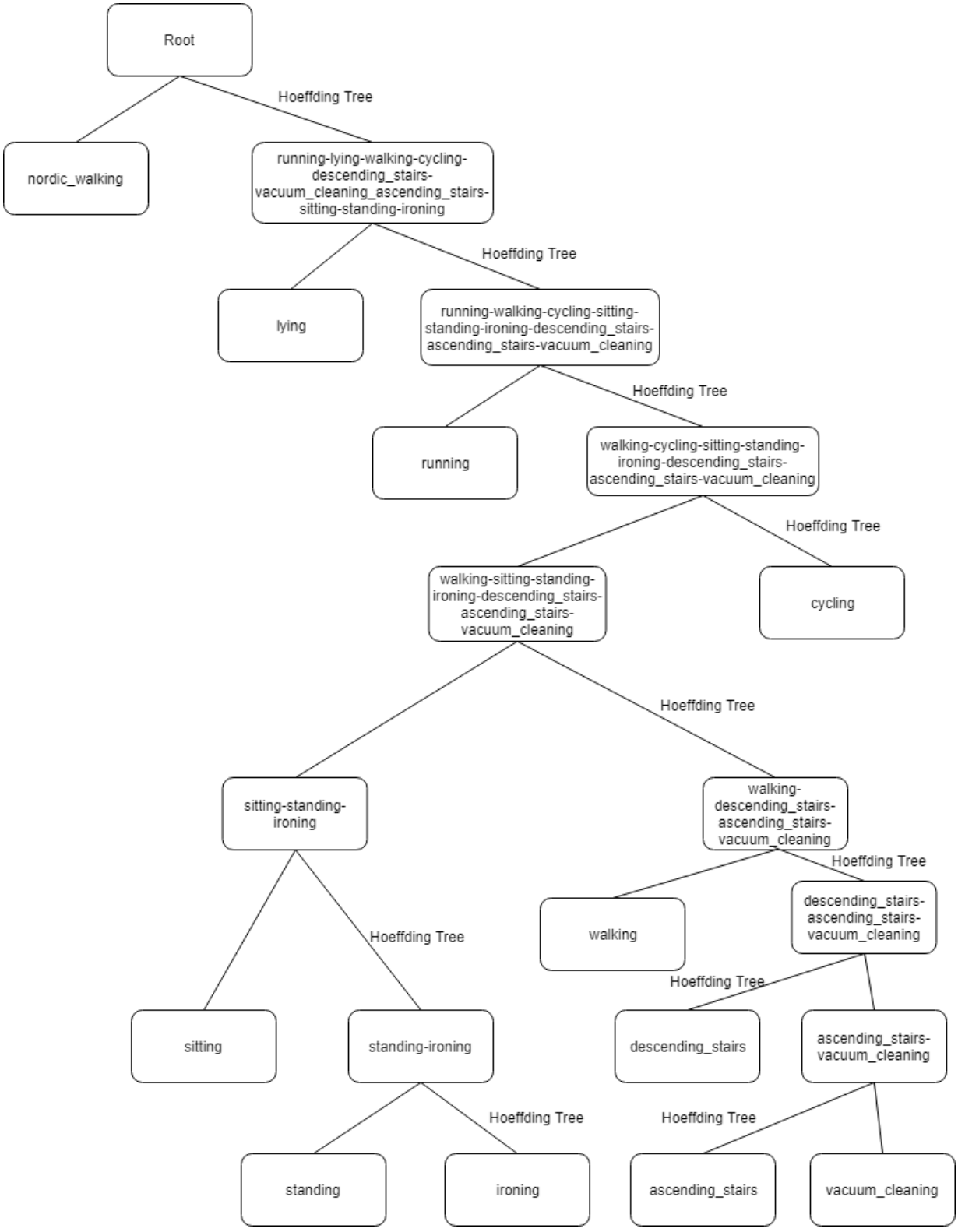

Figure A.8: Tree generated by testing with user 1

Activity	**Total Instances**	**User 1**	**Accuracy**
Lying	227	206	90.7%
Nordic Walking	169	0	0%
Walking	185	168	90.8%
Running	174	135	77.6%
Cycling	197	89	45.2%
Descending Stairs	124	87	70.2%
Ascending Stairs	133	102	76.7%
Vacuum Cleaning	191	127	66.5%
Ironing	196	191	97.4%
Sitting	196	1	0.5%
Standing	181	3	1.7%

Table A.8: Results by testing the tree with user 1

User 1: KNN - 596(28.64%) Naive Bayes - 1786(85.82%), Hoeffding Tree - 1786(85.82%)

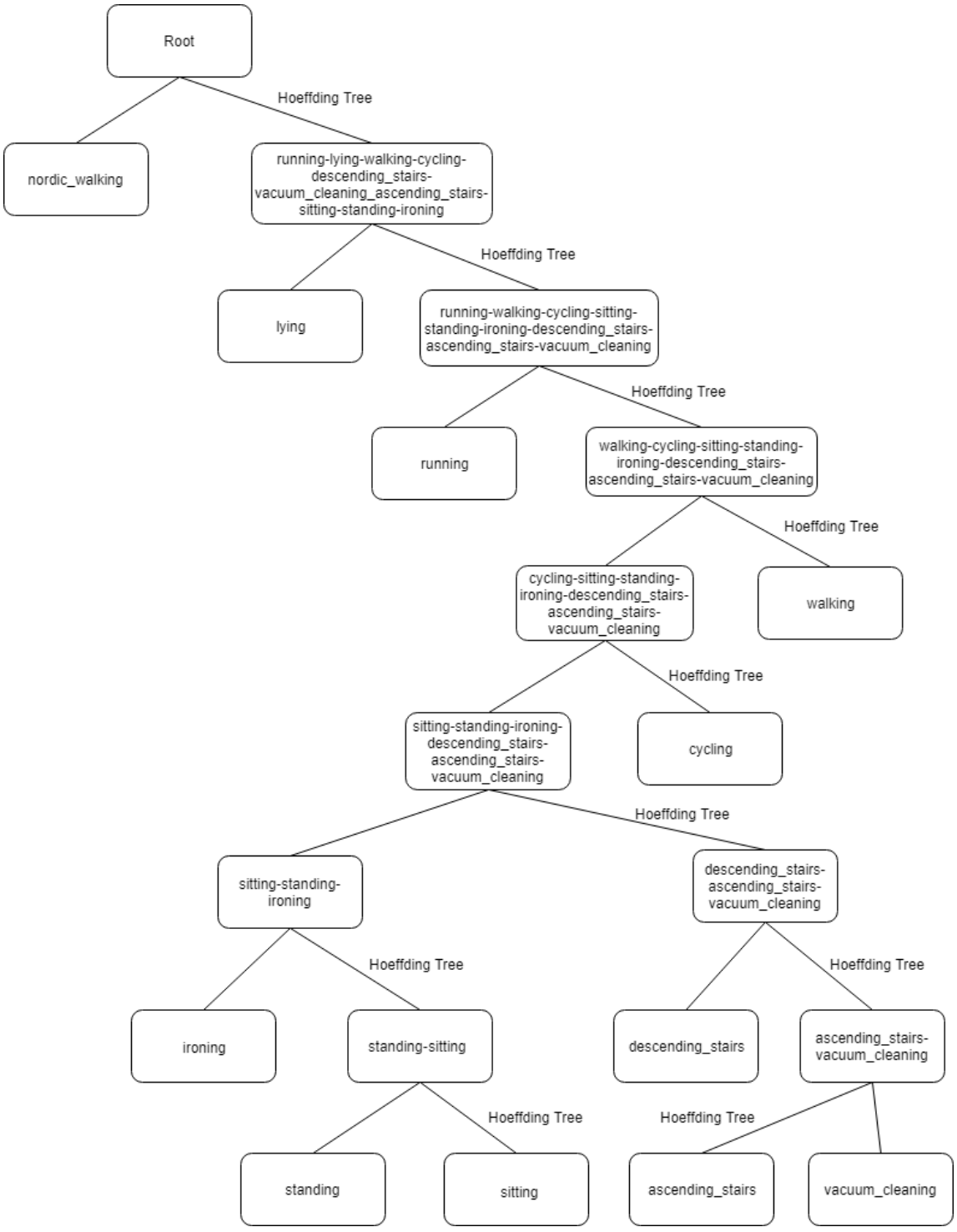

Figure A.9: Tree generated by testing with user 2

Activity	**Total Instances**	**User 2**	**Accuracy**
Lying	196	177	90.3%
Nordic Walking	247	0	0%
Walking	272	196	72.1%
Running	73	70	95.9%
Cycling	210	179	85.2%
Descending Stairs	126	85	67.5%
Ascending Stairs	145	78	53.8%
Vacuum Cleaning	172	97	56.4%
Ironing	241	204	84.6%
Sitting	186	143	76.9%
Standing	213	204	95.8%

Table A.9: Results by testing the tree with user 2

User 2: KNN - 579(27.82%) Naive Bayes - 1756(84.38%), Hoeffding Tree - 1756(84.38%)

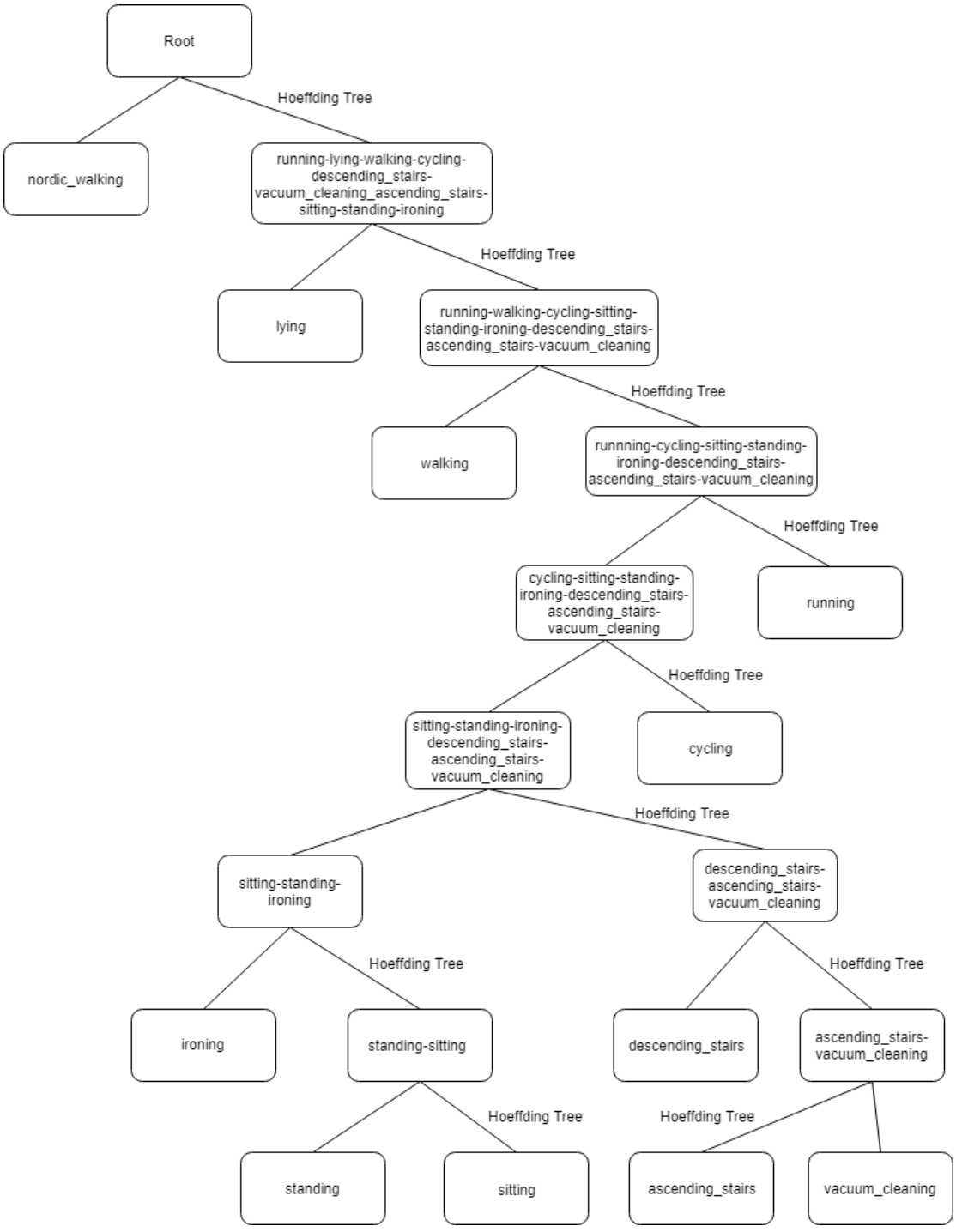

Figure A.10: Tree generated by testing with user 3

User 3: KNN - 586(28.16%) Naive Bayes - 1777(85.39%), Hoeffding Tree - 1777(85.39%)

Activity	**Total Instances**	**User 3**	**Accuracy**
Lying	184	162	88%
Nordic Walking	0	0	Not Count
Walking	239	226	94.6%
Running	0	0	Not Count
Cycling	0	0	Not Count
Descending Stairs	127	103	81.1%
Ascending Stairs	87	43	49.4%
Vacuum Cleaning	169	96	56.8%
Ironing	233	170	73.0%
Sitting	240	45	18.8%
Standing	171	157	91.8%

Table A.10: Results by testing the tree with user 3

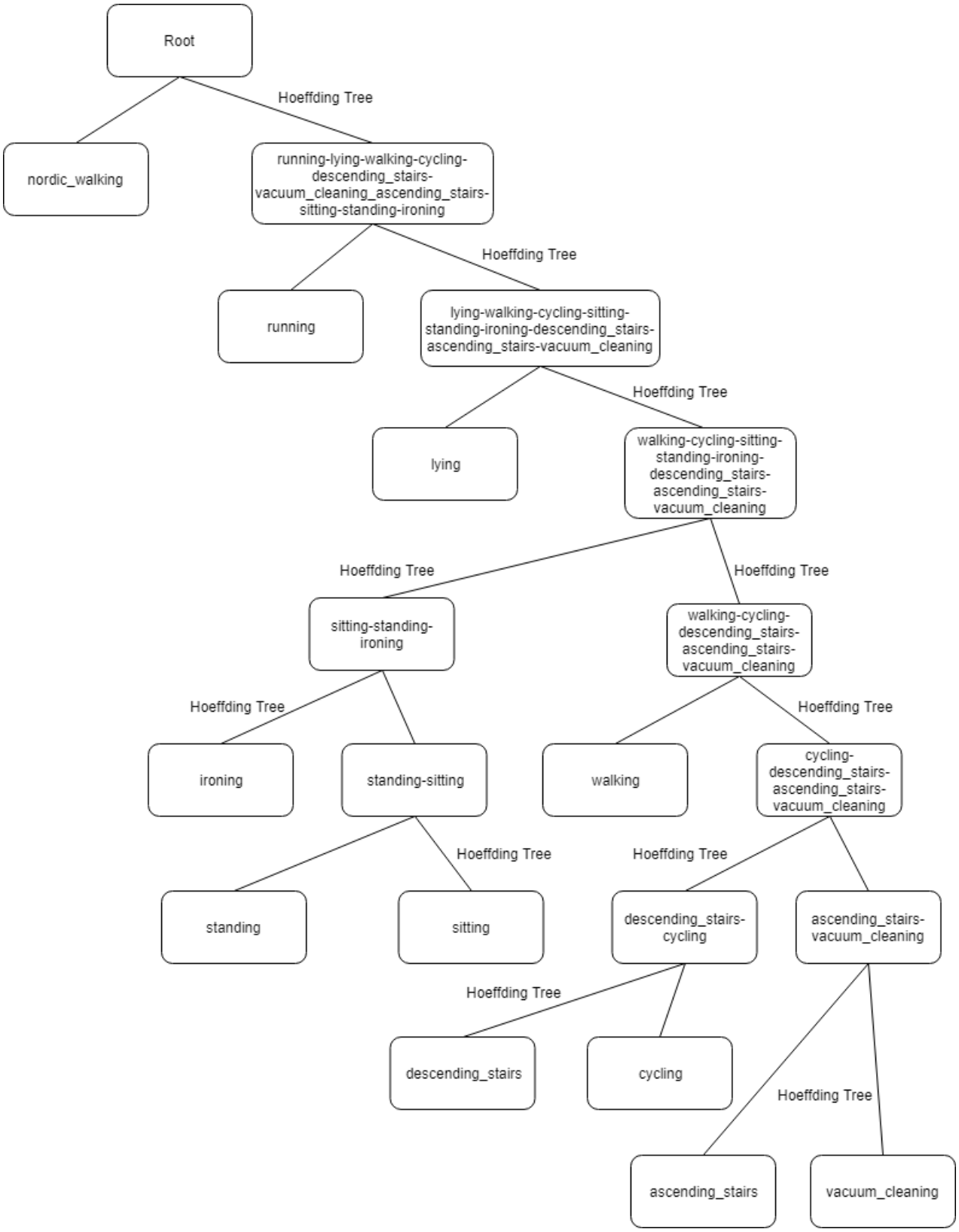

Figure A.11: Tree generated by testing with user 4

Activity	Total Instances	User 4	Accuracy
Lying	193	176	91.2%
Nordic Walking	230	149	64.8%
Walking	266	241	90.6%
Running	0	0	Not Count
Cycling	186	94	50.5%
Descending Stairs	119	60	50.4%
Ascending Stairs	139	82	59.0%
Vacuum Cleaning	167	134	80.2%
Ironing	208	113	54.3%
Sitting	212	25	11.8%
Standing	206	159	77.2%

Table A.11: Results by testing the tree with user 4

User 4: KNN-577(27.73%) Naive Bayes- 1733(83.28%) Hoeffding Tree-1733(83.28%)

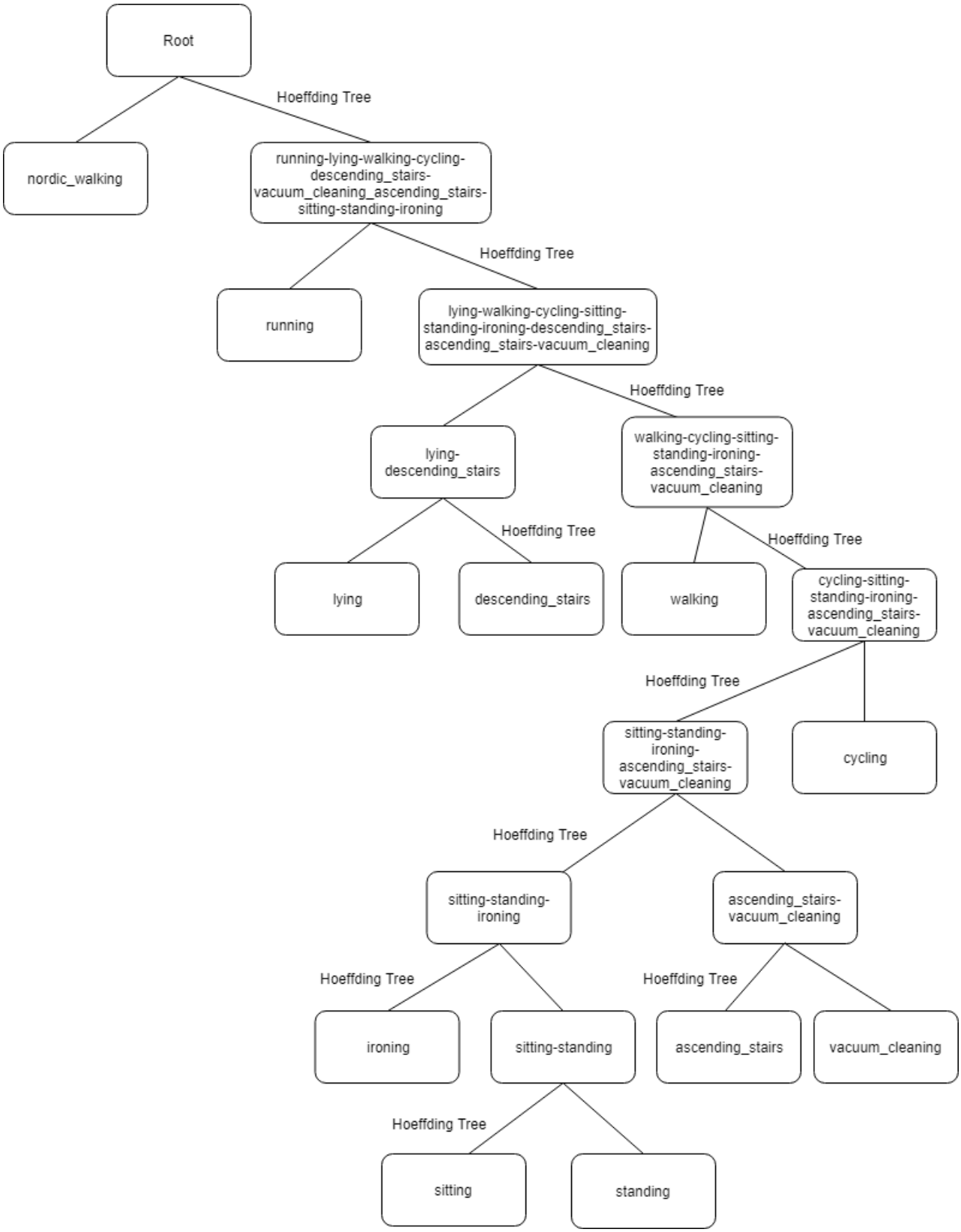

Figure A.12: Tree generated by testing with user 5

User 5: KNN - 606(29.12%), Naive Bayes 1770(85.06%), Hoeffding Tree 1770(85.06%)

Activity	**Total Instances**	**User 5**	**Accuracy**
Lying	198	11	5.6%
Nordic Walking	219	210	95.9%
Walking	267	231	86.5%
Running	202	194	96.0%
Cycling	205	144	70.2%
Descending Stairs	106	59	55.7%
Ascending Stairs	119	103	86.6%
Vacuum Cleaning	203	105	51.7%
Ironing	276	246	89.1%
Sitting	224	63	28.1%
Standing	184	149	80.1%

Table A.12: Results by testing the tree with user 5

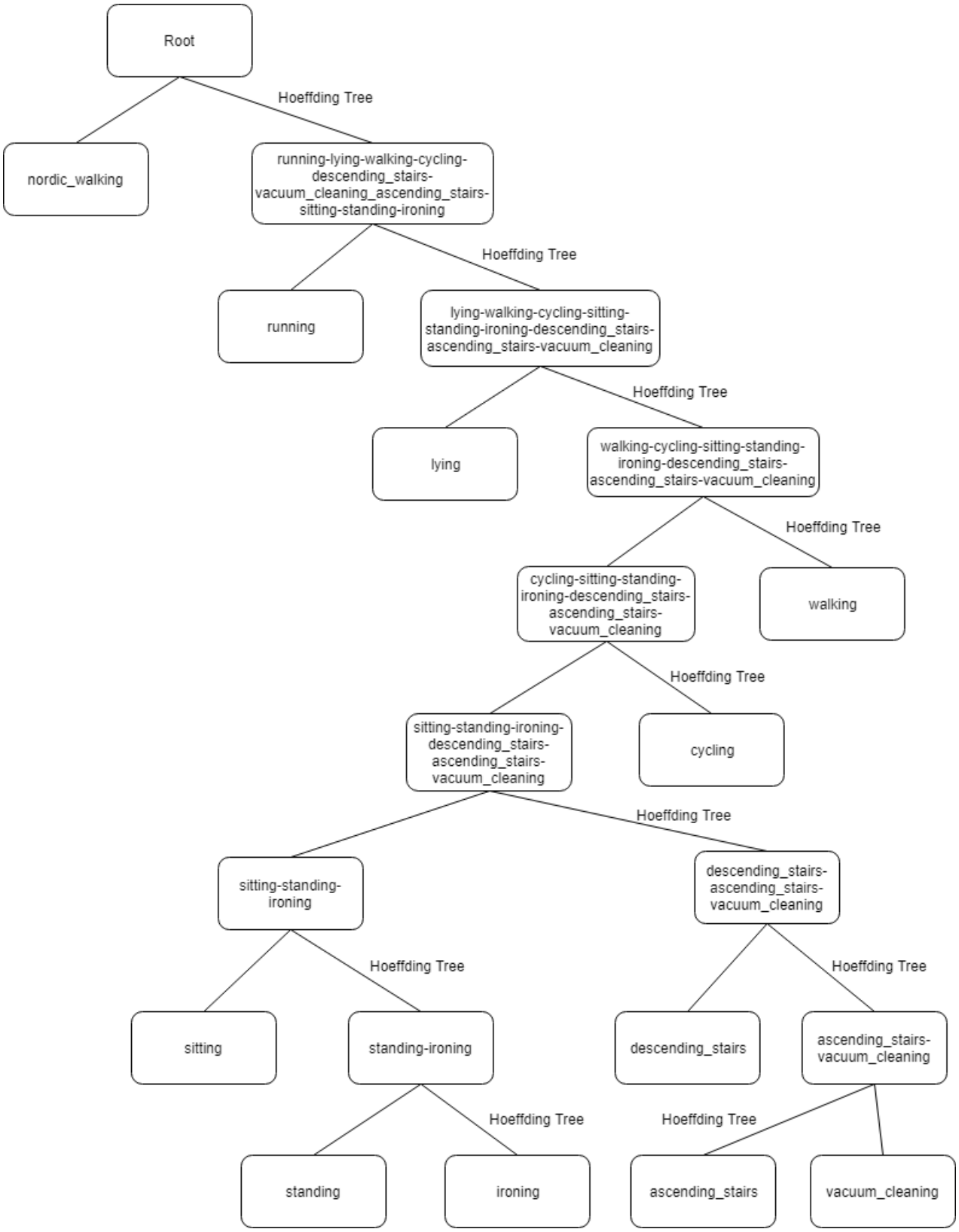

Figure A.13: Tree generated by testing with user 7

User 7: KNN - 588(28.26%), Naive Bayes 1781(85.58%), Hoeffding Tree 1781(85.58%)

Activity	**Total Instances**	**User 7**	**Accuracy**
Lying	214	199	93.0%
Nordic Walking	240	223	93.0%
Walking	281	266	94.7%
Running	27	23	85.2%
Cycling	189	155	82.0%
Descending Stairs	97	81	83.5%
Ascending Stairs	147	86	58.5%
Vacuum Cleaning	179	34	19.0%
Ironing	246	242	98.4%
Sitting	102	36	35.3%
Standing	215	160	74.4%

Table A.13: Results by testing the tree with user 7

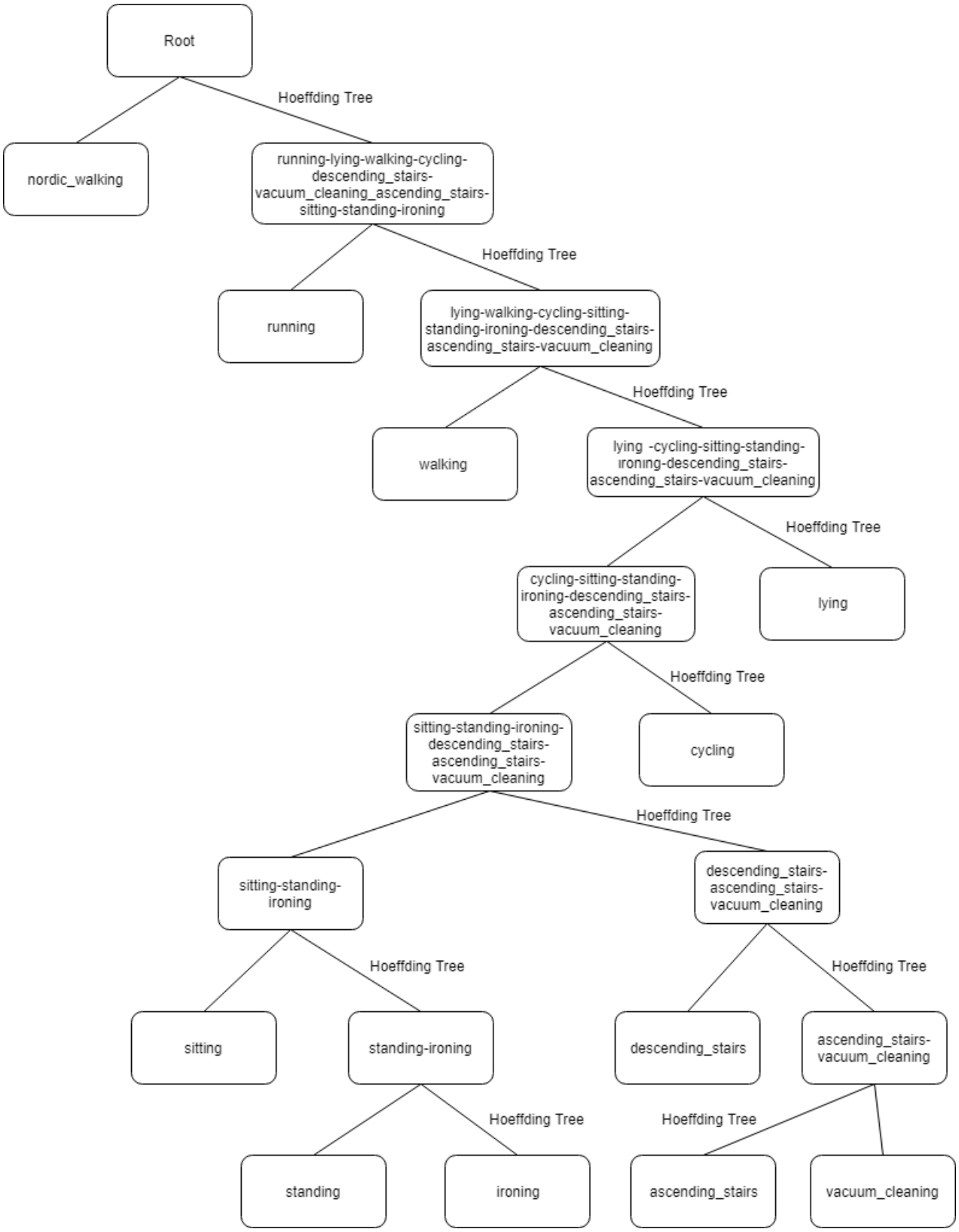

Figure A.14: Tree generated by testing with user 8

User 8 - KNN (78.42%), Naive Bayes (85.68%), Hoeffding Tree (85.68%)

Activity	Total Instances	User 8	Accuracy
Lying	202	6	3.0%
Nordic Walking	241	0	0%
Walking	263	246	93.5%
Running	134	52	38.8%
Cycling	212	157	74.1%
Descending Stairs	80	67	83.8%
Ascending Stairs	98	52	53.1%
Vacuum Cleaning	203	88	43.3%
Ironing	274	227	82.8%
Sitting	191	1	0.5%
Standing	210	174	82.9%

Table A.14: Results by testing the tree with user 8

A.0.3 Multi-Classification Classifiers Results for Second Experience

Activity	Total Instances	User 1	Accuracy
Lying	225	206	91.6%
Nordic Walking	169	57	33.7%
Walking	186	98	52.7%
Running	177	164	92.7%
Cycling	196	181	92.3%
Descending Stairs	125	98	78.4%
Ascending Stairs	131	113	86.3%
Vacuum Cleaning	192	178	92.7%
Ironing	196	193	98.5%
Sitting	196	0	0%
Standing	181	80	44.2%

Table A.15: Results by testing multi-classifier approach with user 1

Activity	Total Instances	User 2	Accuracy
Lying	194	177	91.2%
Nordic Walking	248	0	0%
Walking	271	4	1.5%
Running	77	74	96.1%
Cycling	209	180	86.1%
Descending Stairs	127	96	75.6%
Ascending Stairs	145	130	89.7%
Vacuum Cleaning	172	166	96.5%
Ironing	241	231	95.9%
Sitting	186	132	71.0%
Standing	213	7	3.3%

Table A.16: Results by testing multi-classifier approach with user 2

Activity	Total Instances	User 3	Accuracy
Lying	183	164	89.7%
Nordic Walking	0	0	Not Count
Walking	239	228	95.4%
Running	0	0	Not Count
Cycling	0	0	Not Count
Descending Stairs	128	101	79.0%
Ascending Stairs	86	72	83.7%
Vacuum Cleaning	170	164	96.5%
Ironing	232	213	91.8%
Sitting	236	213	90.3%
Standing	171	129	75.4%

Table A.17: Results by testing multi-classifier approach with user 3

Activity	Total Instances	User 4	Accuracy
Lying	191	178	93.2%
Nordic Walking	229	144	62.9%
Walking	264	250	94.7%
Running	0	0	Not Count
Cycling	188	134	72.3%
Descending Stairs	110	96	87.3%
Ascending Stairs	140	127	90.7%
Vacuum Cleaning	167	155	92.8%
Ironing	208	183	88.0%
Sitting	212	170	80.2%
Standing	206	188	91.3%

Table A.18: Results by testing multi-classifier approach with user 4

Activity	Total Instances	User 5	Accuracy
Lying	196	179	91.3%
Nordic Walking	219	213	97.3%
Walking	267	254	95.1%
Running	206	194	94.2%
Cycling	204	184	90.2%
Descending Stairs	106	78	73.6%
Ascending Stairs	119	110	92.4%
Vacuum Cleaning	204	174	85.3%
Ironing	275	254	92.4%
Sitting	224	148	66.1%
Standing	185	70	37.8%

Table A.19: Results by testing multi-classifier approach with user 5

Activity	Total Instances	User 7	Accuracy
Lying	212	200	94.3%
Nordic Walking	240	233	97.1%
Walking	281	266	94.7%
Running	29	23	79.3%
Cycling	189	176	93.1%
Descending Stairs	97	81	83.5%
Ascending Stairs	146	130	89.0%
Vacuum Cleaning	180	151	83.9%
Ironing	246	192	87.0%
Sitting	103	91	88.3%
Standing	214	214	100%

Table A.20: Results by testing multi-classifier approach with user 7

Activity	Total Instances	User 8	Accuracy
Lying	199	187	94.0%
Nordic Walking	241	0	0%
Walking	263	242	92.0%
Running	138	122	88.4%
Cycling	212	162	76.4%
Descending Stairs	81	38	47.0%
Ascending Stairs	97	70	72.2%
Vacuum Cleaning	202	187	92.6%
Ironing	246	24	9.8%
Sitting	191	169	88.5%
Standing	275	96	35.0%

Table A.21: Results by testing multi-classifier approach with user 8